

Introduction to Computational Quantum Mechanics

Dr. Roman Schmied

Department of Physics, University of Basel, Switzerland
roman.schmied@unibas.ch

*I hear, and I forget.
I see, and I remember.
I do, and I understand.
—Confucius*

Contents

0	outline of this lecture	7
0.1	introduction	7
0.2	what this lecture is <i>not</i> about	8
0.3	Why Mathematica?	8
0.4	outline of discussed topics	8
1	Mathematica language overview	9
1.1	introduction	9
1.1.1	exercises	10
1.2	variables and assignments	11
1.2.1	immediate vs. delayed assignments	11
1.2.2	exercises	12
1.3	four kinds of bracketing	12
1.4	prefix and postfix	13
1.4.1	exercises	14
1.5	programming constructs	14
1.5.1	procedural programming	14
1.5.2	exercises	15
1.5.3	functional programming	15
1.5.4	exercises	16
1.6	function definitions	16
1.6.1	immediate function definitions	16
1.6.2	delayed function definitions	17
1.6.3	functions that remember their results	18
1.6.4	functions with conditions on their arguments	18
1.6.5	fifteen ways to define the factorial function	19
1.6.6	exercises	22
1.7	vectors and matrices	22
1.7.1	vectors	22
1.7.2	matrices	23
1.7.3	sparse vectors and matrices	23
1.7.4	matrix diagonalization	24
1.7.5	exercises	27
1.8	complex numbers	27
1.9	units	28

2	quantum mechanics	31
2.1	basis sets and representations	31
2.1.1	incomplete basis sets	32
2.1.2	exercises	32
2.2	time-independent Schrödinger equation	33
2.2.1	diagonalization	34
2.2.2	exercises	34
2.3	time-dependent Schrödinger equation	34
2.3.1	time-independent basis	35
2.3.2	time-dependent basis: interaction picture	36
2.3.3	special case: $[\hat{\mathcal{H}}(t), \hat{\mathcal{H}}(t')] = 0 \forall(t, t')$	36
2.3.4	special case: time-independent Hamiltonian	37
2.3.5	exercises	37
2.4	basis construction	37
2.4.1	description of a single degree of freedom	37
2.4.2	description of coupled degrees of freedom	38
2.4.3	exercises	40
3	spin systems	41
3.1	quantum-mechanical spin and angular momentum operators	41
3.1.1	exercises	42
3.2	spin-1/2 electron in a dc magnetic field	43
3.2.1	time-independent Schrödinger equation	43
3.2.2	exercises	44
3.3	coupled spin systems: ^{87}Rb hyperfine structure	44
3.3.1	eigenstate analysis	46
3.3.2	“magic” magnetic field	48
3.3.3	coupling to an oscillating magnetic field	49
3.3.4	exercises	53
3.4	coupled spin systems: transverse Ising model	54
3.4.1	basis set	55
3.4.2	asymptotic ground states	55
3.4.3	Hamiltonian diagonalization	56
3.4.4	analysis of the ground state	57
3.4.5	exercises	63
4	real-space systems	65
4.1	one particle in one dimension	65
4.1.1	basis functions	65
4.1.2	example: square well with bottom step	70
4.1.3	dynamics	76
4.2	non-linear Schrödinger equation	80
4.2.1	ground state of the non-linear Schrödinger equation	81
4.3	several particles in one dimension: interactions	84
4.3.1	two particles in one dimension with contact interaction	84
4.3.2	two particles in one dimension with arbitrary interaction	87
4.4	one particle in several dimensions	88

5 combining space and spin	93
5.1 one particle with spin in one dimension	93
5.1.1 separable Hamiltonian	93
5.1.2 non-separable Hamiltonian	94
5.1.3 exercises	100
6 path-integral methods	101
6.1 path integrals for propagation in time	101
6.2 path integrals for propagation in imaginary time	105
6.2.1 example: a single particle in a 1D harmonic oscillator	106
6.3 Monte Carlo integration	108
6.3.1 one-dimensional uniform integration	108
6.3.2 one-dimensional integration with weight	110
6.3.3 the Metropolis–Hastings algorithm	112
6.4 Path-Integral Monte Carlo	115
6.4.1 calculating the density	118
Index	123

Chapter 0

outline of this lecture

This document is the lecture script of a one-semester course taught at the University of Basel in the Fall semesters of 2012 and 2013. It is aimed at advanced students of physics who are familiar with the concepts and notations of quantum mechanics.

0.1 introduction

Quantum mechanics lectures can often be separated into two classes. In the first class you get to know Schrödinger's equation and find the form and dynamics of simple physical systems (square well, harmonic oscillator, hydrogen atom); most calculations are analytic and inspired by calculations originally done in the 1920s and 1930s. In the second class you learn about large systems such as molecular structures, crystalline solids, or lattice models; these calculations are usually so complicated that it is difficult for the student to understand them in all detail.

This lecture tries to bridge the gap between simple analytic calculations and brute-force large-scale computations. We will revisit most of the problems encountered in introductory quantum mechanics, focusing on computer implementations for finding analytical as well as numerical solutions and their visualization. We will be approaching topics such as the following:

- You have calculated the energy eigenstates of single particles in simple potentials. How can such calculations be generalized to non-trivial potentials?
- How can we calculate the behavior of interacting particles?
- How can we describe the internal spin structure of particles? How does this internal structure couple to the particles' motion?
- You have heard that quantum mechanics describes our everyday world just as well as classical mechanics does, but you may never have seen an example where this is calculated in detail and where the transition from the classical behavior to the quantum-mechanical behavior is evident.

Most of these calculations are too complicated to be done by hand. Even relatively simple problems, such as two interacting particles in a one-dimensional trap, do not have analytic solutions and require the use of computers for their solution and visualization. More complex problems scale exponentially with the number of degrees of freedom, and make the use of large computer simulations unavoidable.

0.2 what this lecture is *not* about

This course is not about *quantum computing*. Quantum computing refers to the proposed use of quantum-mechanical entanglement of physical systems for speeding up certain calculations, such as factoring large numbers.

This course is not about *large-scale quantum calculations* such as solid-state physics or quantum chemistry. It will, however, provide you with the tools for understanding such large-scale calculations better.

0.3 Why Mathematica?

This course will be taught in the Mathematica programming language.¹ No prior knowledge of Mathematica is necessary, and Mathematica licenses will be provided. Alternatives to Mathematica, such as Matlab or Maple, may be used by the students, but only limited help will be available from the instructor.

There are many reasons for choosing Mathematica over other computer-algebra systems (CAS):

- Mathematica is a very high-level programming environment, which allows the user to focus on *what* he wants to do instead of *how* it is done. A very large number of algorithms for analytic and numerical calculations is included in the Mathematica kernel and its libraries.
- Mathematica seamlessly mixes analytic and numerical facilities. For many calculations it allows you to push analytic evaluations as far as possible, and then continue with numerical evaluations by making only trivial changes.
- Mathematica supports a wide range of programming paradigms, which means that you can keep programming in your favorite style. See section 1.6.5 for a concrete example.
- The instructor is more familiar with Mathematica than any other CAS.

0.4 outline of discussed topics

Chapter 1 gives an introduction to Mathematica, with a focus on techniques that will be useful for this lecture.

Chapter 2 makes the connection between quantum mechanics and the vector/matrix calculus of Mathematica.

Chapter 3 discusses systems with finite-dimensional Hilbert spaces, focusing on spin systems.

Chapter 4 discusses the quantum mechanics of particles moving in one- and several-dimensional space.

Chapter 5 connects the topics of chapters 3 and 4.

Chapter 6 presents a brief introduction to path integrals and Monte Carlo integration.

¹At the time of writing, Mathematica version 9 was being used.

Chapter 1

Mathematica language overview

If you have little or no experience with Mathematica, you may start here:

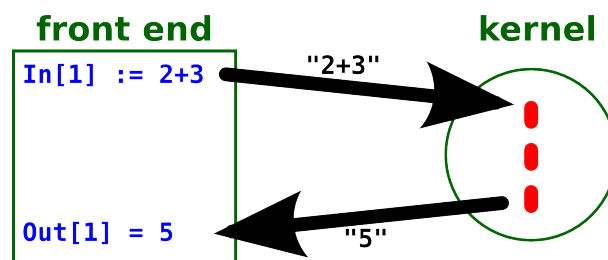
<http://www.wolfram.com/support/learn/higher-education.html>

Further useful links:

- <http://reference.wolfram.com/mathematica/guide/LanguageOverview.html>
- <http://reference.wolfram.com/mathematica/guide/Mathematica.html>

1.1 introduction

Mathematica is an interactive language for mathematical calculations. The Mathematica system is composed of two main components: the *front end*, where you write the input, give execution commands, and see the output, and the *kernel*, which does the actual calculations.



This distinction is important to remember because the kernel remembers all the operations in the order they are sent to it, and this order may have nothing to do with the order in which these commands are displayed in the front end.

When you start Mathematica you see an empty “notebook” in which you can write commands. These commands are written in a mixture of text and mathematical symbols and structures, and it takes a bit of practice to master all the special input commands. In the beginning you can write all your input in pure text mode, if you prefer. Let’s try an example: add the numbers $2 + 3$ by giving the input

```
1 In[1]:= 2+3
```

and, with the cursor anywhere within the “cell” containing this text (look on the right edge of the notebook to see cell limits and groupings) you press “shift-enter”. This sends the contents of this cell to the kernel, which executes it and returns a result that is displayed in the next cell:

```
1 Out[1]= 5
```

If there are many input cells in a notebook, they only get executed in order if you select “Evaluate Notebook” from the “Evaluation” menu; otherwise you can execute the input cells in any order you wish by simply setting the cursor within one cell and pressing “shift-enter”.

1.1.1 exercises

Do the following calculations in Mathematica, and try to understand their structure:

Q1.1 Calculate the numerical value of $\zeta(3)$ with

```
1 In[2]:= N[Zeta[3]]
```

Q1.2 Square the previous result (%) with

```
1 In[3]:= %^2
```

Q1.3 Calculate $\int_0^\infty \sin(x)e^{-x}dx$ with

```
1 In[4]:= Integrate[Sin[x] Exp[-x], {x, 0, Infinity}]
```

Q1.4 Calculate the first 1000 digits of π with

```
1 In[5]:= N[Pi, 1000]
```

Q1.5 Calculate the Clebsch–Gordan coefficient $\langle 1000, 100; 2000, -120 | 1100, -20 \rangle$:

```
1 In[6]:= ClebschGordan[{1000, 100}, {2000, -120}, {1100, -20}]
```

Q1.6 Calculate the limit $\lim_{x \rightarrow 0} \frac{\sin x}{x}$ with

```
1 In[7]:= Limit[Sin[x]/x, x -> 0]
```

Q1.7 Make a plot of the above function with

```
1 In[8]:= Plot[Sin[x]/x, {x, -20, 20}, PlotRange -> All]
```

Q1.8 Draw a Mandelbrot set with

```

1 In[9]:= F[c_, imax_] := Abs[NestWhile[#^2 + c &, 0.,
2       Abs[#] <= 2 &, 1, imax]] <= 2
3 In[10]:= With[{n = 100, imax = 1000},
4       Graphics[Raster[Table[Boole[!F[x + I y, imax]],
5       {y, -2, 2, 1/n}, {x, -2, 2, 1/n}]]]]

```

1.2 variables and assignments

<http://reference.wolfram.com/mathematica/howto/WorkWithVariablesAndFunctions.html>

Variables in Mathematica can be letters or words with uppercase or lowercase letters, including Greek symbols. Assigning a value to a variable is done with the `=` symbol,

```

1 In[11]:= a = 5
2 Out[11]= 5

```

If you wish to suppress the output, then you must end the command with a semi-colon:

```

1 In[12]:= a = 5;

```

The variable name can then be used anywhere in an expression:

```

1 In[13]:= a + 2
2 Out[13]= 7

```

1.2.1 immediate vs. delayed assignments

<http://reference.wolfram.com/mathematica/tutorial/ImmediateAndDelayedDefinitions.html>

Consider the two commands

```

1 In[14]:= a = RandomReal[]
2 Out[14]= 0.38953
3 In[15]:= b := RandomReal[]

```

(your random number will be different).

The first statement `a = ...` is an *immediate assignment*, which means that its right-hand side is evaluated when you press shift-enter, produces a specific random value, and is assigned to the variable `a` (and printed out). From now on, every time you use the variable `a`, the *exact same* number will be substituted. In this sense, the variable `a` contains the number 0.38953 and has no memory of where it got this number from. You can check the definition of `a` with `?a`:

```

1 In[16]:= ?a
2      Global`a
3      a = 0.38953

```

The definition `b:=...` is a *delayed assignment*, which means that when you press shift-enter the right-hand side is not evaluated but merely stored as a definition of `b`. From now on, every time you use the variable `b`, its right-hand-side definition will be substituted and executed, resulting in a new random number each time. You can check the definition of `b` with

```
1 In[17]:= ?b
2          Global`b
3          b := RandomReal []
```

Let's compare the repeated performance of `a` and `b`:

```
1 In[18]:= {a, b}
2 Out[18]= {0.38953, 0.76226}
3 In[19]:= {a, b}
4 Out[19]= {0.38953, 0.982921}
5 In[20]:= {a, b}
6 Out[20]= {0.38953, 0.516703}
7 In[21]:= {a, b}
8 Out[21]= {0.38953, 0.0865169}
```

1.2.2 exercises

Q1.9 Explain the difference between

```
1 In[22]:= x = u + v
```

and

```
1 In[23]:= y := u + v
```

In particular, distinguish the cases where `u` and `v` are already defined before `x` and `y` are defined, where they are defined only afterwards, and where they are defined before but change values after the definition of `x` and `y`.

1.3 four kinds of bracketing

<http://reference.wolfram.com/mathematica/tutorial/TheFourKindsOfBracketingInMathematica.html>

There are four types of brackets in Mathematica:

- parentheses for grouping, for example in mathematical expressions:

```
1 In[24]:= 2*(3-7)
```

- square brackets for function calls:

```
1 In[25]:= Sin[0.2]
```

- curly braces for lists:

```
1 In[26]:= v = {a, b, c}
```

- double square brackets for indexing within lists: (see section 1.7)

```
1 In[27]:= v[[2]]
```

1.4 prefix and postfix

There are several ways of evaluating a function call in Mathematica, and we will see most of them in this lecture. As examples of function calls with a single argument, the main ways in which $\sin(0.2)$ and $\sqrt{2+3}$ can be calculated are

standard notation (infinite precedence):

```
1 In[28]:= Sin[0.2]
2 Out[28]= 0.198669
3 In[29]:= Sqrt[2+3]
4 Out[29]= Sqrt[5]
```

prefix notation with @ (quite high precedence, higher than multiplication):

```
1 In[30]:= Sin @ 0.2
2 Out[30]= 0.198669
3 In[31]:= Sqrt @ 2+3
4 Out[31]= 3+Sqrt[2]
```

Notice how the high precedence of the @ operator effectively evaluates $(\text{Sqrt} @ 2) + 3$, not $\text{Sqrt} @ (2 + 3)$.

postfix notation with // (quite low precedence, lower than addition):

```
1 In[32]:= 0.2 // Sin
2 Out[32]= 0.198669
3 In[33]:= 2+3 // Sqrt
4 Out[33]= Sqrt[5]
```

Notice how the low precedence of the // operator effectively evaluates $(2+3) // N$, not $2 + (3 // N)$.

Postfix notation is often used to transform the output of a calculation:

- Adding `//N` to the end of a command will convert the result to decimal representation, if possible.
- Adding `//MatrixForm` to the end of a matrix calculation will display the matrix in a tabular form.
- Adding `//Timing` to the end of a calculation will display the result together with the amount of time it took to execute.

If you are not sure which form is appropriate, for example if you don't know the precedence of the involved operations, then you should use the standard notation or place parentheses where needed.

1.4.1 exercises

Q1.10 Calculate the decimal value of Euler's constant e (**E**) using standard, prefix, and postfix notation.

1.5 programming constructs

When you program in Mathematica you can choose between a number of different programming paradigms, and you can mix these as you like. Depending on the chosen style, your program may run much faster or much slower.

1.5.1 procedural programming

<http://reference.wolfram.com/mathematica/guide/ProceduralProgramming.html>

A subset of Mathematica behaves very similarly to C, Python, Java, or other procedural programming languages. Be very careful to distinguish semi-colons, which separate commands within a single block of code, from commas, which separate different code blocks!

Looping constructs behave like in common programming languages:

```
1 In[34]:= For[i = 1, i <= 10, i++,
2         Print[i]]
```

```
1 In[35]:= Do[Print[i], {i, 1, 10}]
```

```
1 In[36]:= i = 1;
2         While[i <= 10,
3             Print[i];
4             i++]
```

Conditional execution: notice that the **If** statement has a return value, similar to the “?” statement of C and Java.

```
1 In[37]:= If[5! > 100,
2         Print["larger"],
3         Print["smaller or equal"]]
```

```
1 In[38]:= If[5! > 100, 1, -1]
2 Out[38]= 1
```

Modularity: code can use local variables within a *module*:

```

1 In[39]:= Module[{i},
2           i = 1;
3           While[i > 1/192, i = i/2];
4           i]
5 Out[39]= 1/256

```

After the execution of this code, the variable `i` is still undefined in the global context.

1.5.2 exercises

- Q1.11** Write a program which sums all integers from 123 to 9968. Use only local variables.
- Q1.12** Write a program which sums consecutive integers, starting from 123, until the sum is larger than 10000. Return the largest integer in this sum. Use only local variables.

1.5.3 functional programming

<http://reference.wolfram.com/mathematica/guide/FunctionalProgramming.html>

Functional programming is a very powerful programming technique which can give large speedups in computation because it can often be parallelized over many computers or CPUs. In our context, we often use lists (vectors or matrices, see section 1.7) and want to apply functions to each one of their elements.

The most common functional programming constructs are

Anonymous functions:¹ you can quickly define a function with parameters `#1=#`, `#2=##`, etc., terminated with the `&` symbol:

```

1 In[40]:= f = #^2 &;
2 In[41]:= f[7]
3 Out[41]= 49
4 In[42]:= g = #1-#2 &;
5 In[43]:= g[88, 9]
6 Out[43]= 79

```

Anonymous functions, for example `#^2&`, are objects just like numbers, matrices, etc. You can assign them to variables, as above; you can also use them directly as arguments to other functions, as below.

Map /@: apply a function to each element of a list.

```

1 In[44]:= a = {1, 2, 3, 4, 5, 6, 7, 8};
2 In[45]:= Map[#^2 &, a]
3 Out[45]= {1, 4, 9, 16, 25, 36, 49, 64}
4 In[46]:= #^2 & /@ a
5 Out[46]= {1, 4, 9, 16, 25, 36, 49, 64}

```

¹see http://en.wikipedia.org/wiki/Anonymous_functions.

Notice how we have used the anonymous function `#^2&` here without ever giving it a name.

Apply @@: apply a function to an entire list and generate a single result. For example, applying `Plus` to a list will calculate the sum of the list elements; applying `Times` will calculate their product. This operation is also known as *reduce*.²

```

1 In[47]:= a = {1, 2, 3, 4, 5, 6, 7, 8};
2 In[48]:= Apply[Plus, a]
3 Out[48]= 36
4 In[49]:= Plus @@ a
5 Out[49]= 36
6 In[50]:= Apply[Times, a]
7 Out[50]= 40320
8 In[51]:= Times @@ a
9 Out[51]= 40320

```

1.5.4 exercises

Q1.13 Write an anonymous function with three arguments, which returns the product of these arguments.

Q1.14 Given a list

```

1 In[52]:= a = {0.1, 0.9, 2.25, -1.9};

```

calculate $x \mapsto \sin(x^2)$ for each element of `a` using the `Map` operation.

Q1.15 Calculate the sum of all the results of **Q1.14**.

1.6 function definitions

<http://reference.wolfram.com/mathematica/tutorial/DefiningFunctions.html>

Functions are assignments (see section 1.2) with parameters. As for parameter-free assignments we distinguish between *immediate* and *delayed* function definitions.

1.6.1 immediate function definitions

We start with *immediate* definitions: a function $f(x) = \sin(x)/x$ is defined with

```

1 In[53]:= f[x_] = Sin[x]/x;

```

Notice the underscore `_` symbol after the variable name `x`: this underscore indicates a *pattern* (denoted by `_`) named `x`, not the symbol `x` itself. Whenever this function `f` is called with any parameter value, this parameter value is inserted wherever `x` appears on the right-hand side, as is expected for a function definition. You can find out how `f` is defined with the `?` operator:

²See <http://en.wikipedia.org/wiki/MapReduce>.


```

1 In[54]:= ?f
2           Global`f
3           f[x_] = Sin[x]/x

```

and you can ask for a function evaluation with

```

1 In[55]:= f[0.3]
2 Out[55]= 0.985067
3 In[56]:= f[0]
4           Power::infy : Infinite expression 1/0 encountered.
5           Infinity::indet : Indeterminate expression 0 ComplexInfinity encountered.
6 Out[56]= Indeterminate

```

Apparently the function cannot be evaluated for $x = 0$. We can fix this by defining a special function value:

```

1 In[57]:= f[0] = 1;

```

Notice that there is no underscore on the left-hand side, so there is no pattern definition. The full definition of `f` is now

```

1 In[58]:= ?f
2           Global`f
3           f[0] = 1
4           f[x_] = Sin[x]/x

```

If the function `f` is called, then these definitions are checked in order of appearance in this list. For example, if we ask for `f[0]`, then the first entry matches and the value 1 is returned. If we ask for `f[0.3]`, then the first entry does not match (since 0 and 0.3 are not strictly equal), but the second entry matches since basically anything can be plugged into the pattern named `x`. The result is $\sin(0.3)/0.3 = 0.985067$, which is what we expected.

1.6.2 delayed function definitions

Just like with delayed assignments (section 1.2.1), we can define delayed function calls. For comparison, we define the two functions

```

1 In[59]:= g1[x_] = x + RandomReal[]
2 Out[59]= 0.949868 + x
3 In[60]:= g2[x_] := x + RandomReal[]

```

Check their effective definitions with `?g1` and `?g2`, and notice that the definition of `g1` was executed immediately when you pressed shift-enter and its result assigned to the function `g1` (with a specific value for the random number, as printed out), whereas the definition of `g2` was left unevaluated and is executed each time anew when you use the function `g2`:

```

1 In[61]:= {g1[2], g2[2]}
2 Out[61]= {2.94987, 2.33811}
3 In[62]:= {g1[2], g2[2]}
4 Out[62]= {2.94987, 2.96273}
5 In[63]:= {g1[2], g2[2]}
6 Out[63]= {2.94987, 2.18215}

```

1.6.3 functions that remember their results

<http://reference.wolfram.com/mathematica/tutorial/FunctionsThatRememberValuesTheyHaveFound.html>

When we define a function that takes a long time to evaluate, we may wish to store its output values such that if the function is called with identical parameter values again, then we do not need to re-evaluate the function but can simply return the already calculated result. We can make use of the interplay between patterns and values, and between immediate and delayed assignments, to construct such a function which remembers its values from previous function calls.

See if you can understand the following definition.

```

1 In[64]:= F[x_] := F[x] = x^7

```

If you ask for `?F` then you will simply see this definition. Now call

```

1 In[65]:= F[2]
2 Out[65]= 128

```

and ask for `?F` again. You see that the specific immediate definition of `F[2]=128` was added to the list of definitions, with the evaluated result 128 (which may have taken a long time to calculate in a more complicated function). The next time you call `F[2]`, the specific definition of `F[2]` will be found earlier in the definitions list than the general definition `F[x_]` and therefore the precomputed value of `F[2]` will be returned.

When you re-define the function `F` after making modifications to it, you must clear the associated remembered values in order for them to be re-computed at the next occasion. It is a good practice to prefix every definition of a self-remembering function with a `Clear` command:

```

1 In[66]:= Clear[F];
2 In[67]:= F[x_] := F[x] = x^9

```

1.6.4 functions with conditions on their arguments

<http://reference.wolfram.com/mathematica/guide/Patterns.html>

Mathematica contains a powerful pattern language that we can use to define functions which only accept certain arguments. For function definitions we will use three main types of patterns:

Anything-goes: A function defined as

```
1 In[68]:= f[x_] := x^2
```

can be called with any sort of arguments, since the pattern `x_` can match *anything*:

```
1 In[69]:= f[4]
2 Out[69]= 16
3 In[70]:= f[2.3-0.1I]
4 Out[70]= 5.28-0.46I
5 In[71]:= f[{1,2,3,4}]
6 Out[71]= {1,4,9,16}
7 In[72]:= f[y^2]
8 Out[72]= y^4
```

Type-restricted: A pattern like `x_Integer` will match only arguments of integer type. If the function is called with a non-matching argument, then the function is not executed:

```
1 In[73]:= g[x_Integer] := x-3
2           g[x_Real] := x+3
3 In[74]:= g[7]
4 Out[74]= 4
5 In[75]:= g[7.1]
6 Out[75]= 10.1
7 In[76]:= g[2+3I]
8 Out[76]= g[2+3I]
```

Conditional: Complicated conditions can be specified with the `/;` operator:

```
1 In[77]:= h[x_/;x<=3] := x^2
2           h[x_/;x>3] := x-11
3 In[78]:= h[2]
4 Out[78]= 4
5 In[79]:= h[5]
6 Out[79]= -6
```

Conditions involving a single function call returning a Boolean value, for example `x_/;PrimeQ[x]`, can be abbreviated with `x_?PrimeQ`.

1.6.5 fifteen ways to define the factorial function

The Wolfram demo <http://www.wolfram.com/training/videos/EDU002/> lists fifteen ways how to define the factorial function. Try to understand as many of these definitions as possible. What this means in practice is that for most problems you can pick the programming paradigm which suits your way of thinking best, instead of being forced into one way or another. The different paradigms have different advantages and disadvantages, which may become clearer to you as you become more familiar with them.

You must call `Clear[f]` between different definitions!

1. Define the function `f` to be an alias of the built-in function `Factorial`: calling `f[5]` is now strictly the same thing as calling `Factorial[5]`, which in turn is the same thing as calling `5!`.

```
1 In[80]:= f = Factorial;
```

2. A call to `f` is forwarded to the function “`!`”: calling `f[5]` triggers the evaluation of `5!`.

```
1 In[81]:= f[n_] := n!
```

3. Use the mathematical definition $n! = \Gamma(n + 1)$:

```
1 In[82]:= f[n_] := Gamma[n+1]
```

4. Use the mathematical definition $n! = \prod_{i=1}^n i$:

```
1 In[83]:= f[n_] := Product[i, {i, n}]
```

5. Rule-based recursion, using Mathematica's built-in pattern-matching capabilities: calling `f[5]` leads to a call of `f[4]`, which leads to a call of `f[3]`, and so on until `f[1]` immediately returns the result 1, after which the program unrolls the recursion stack and does the necessary multiplications:

```
1 In[84]:= f[1] = 1;
2         f[n_] := n f[n-1]
```

6. The same recursion but without rules (no pattern-matching):

```
1 In[85]:= f[n_] := If[n == 1, 1, n f[n-1]]
```

7. Define the same recursion defined through functional programming: `f` is a function whose name is `#0` and whose first (and only) argument is `#1`. The end of the function definition is marked with `&`.

```
1 In[86]:= f = If[#1 == 1, 1, #1 #0[#1-1]]&;
```

8. procedural programming with a `Do` loop:

```
1 In[87]:= f[n_] := Module[{t = 1},
2         Do[t = t i, {i, n}];
3         t]
```

9. procedural programming with a `For` loop: this is how you would compute factorials in procedural programming languages like C. It is a very precise step-by-step prescription of how exactly the computer is supposed to do the calculation.

```

1 In[88]:= f[n_] := Module[{t = 1, i},
2       For[i = 1, i <= n, i++,
3           t *= i];
4       t]

```

10. Make a list of the numbers $1 \dots n$ (with `Range[n]`) and then multiply them together at once, by applying the function `Times` to this list. This is the most elegant way of multiplying all these numbers together, because both the generation of the list of integers and their multiplication are done with internally optimized methods. The programmer merely specifies *what* he would like the computer to do, and not *how* it is to be done.

```

1 In[89]:= f[n_] := Times @@ Range[n]

```

11. Make a list of the numbers $1 \dots n$ and then multiply them together one after the other.

```

1 In[90]:= f[n_] := Fold[Times, 1, Range[n]]

```

12. Functional programming: make a list of functions $\{t \mapsto t, t \mapsto 2t, t \mapsto 3t, \dots, t \mapsto nt\}$, and then, starting with the number 1, apply each of these functions once.

```

1 In[91]:= f[n_] := Fold[#2[#1]&, 1,
2       Array[Function[t, #1 t]&, n]]

```

13. Construct a list whose length we know to be $n!$:

```

1 In[92]:= f[n_] := Length[Permutations[Range[n]]]

```

14. Use repeated pattern-based replacement (`//.`) to find the factorial: start with the object $\{1, n\}$ and apply the given rule until the result no longer changes because the pattern no longer matches.

```

1 In[93]:= f[n_] := First[{1,n} //. {a_,b_;/;b>0} :> {b a,b-1}]

```

15. Build a string whose length is $n!$:

```

1 In[94]:= f[n_] := StringLength[Fold[
2       StringJoin[Table[#1, {#2}]]&,
3       "A", Range[n]]]

```

1.6.6 exercises

- Q1.16** In which ones of the definitions of section 1.6.5 can you replace a delayed assignment (`:=`) with an immediate assignment (`=`) or vice-versa? What changes if you do this replacement?
- Q1.17** Can you use the trick of section 1.6.3 for any of the definitions of section 1.6.5?
- Q1.18** Write two very different programs which calculate the first hundred Fibonacci numbers $\{1, 1, 2, 3, 5, 8, \dots\}$, where each number is the sum of the two preceding ones.

1.7 vectors and matrices

In this lecture we will use vectors and matrices to represent quantum states and operators, respectively.

1.7.1 vectors

<http://reference.wolfram.com/mathematica/tutorial/VectorOperations.html>

In Mathematica, vectors are represented as lists of objects, for example lists of real or complex numbers:

```
1 In[95]:= v = {1,2,3,2,1,7+I};
2 In[96]:= Length[v]
3 Out[96]= 6
```

You can access any element by its index, using double brackets, with the first element having index 1 (as in Fortran or Matlab), *not* 0 (as in C, Java, or Python):

```
1 In[97]:= v[[4]]
2 Out[97]= 2
```

Negative indices count from the end of the list:

```
1 In[98]:= v[[-1]]
2 Out[98]= 7+I
```

Lists can contain arbitrary elements (for example strings, graphics, expressions, lists, functions, etc.).

If two vectors \vec{a} and \vec{b} of equal length are defined, then their scalar product $\vec{a}^* \cdot \vec{b}$ is calculated with

```
1 In[99]:= a = {0.1, 0.2, 0.3 + 2 I};
2 In[100]:= b = {-0.27 I, 0, 2};
3 In[101]:= Conjugate[a].b
4 Out[101]= 0.6 - 4.027 I
```

Vectors can be element-wise added, subtracted, multiplied etc. with the usual operators:

```

1 In[102]:= a + b
2 Out[102]={0.1 - 0.27 I, 0.2, 2.3 + 2. I}
3 In[103]:= 2 a
4 Out[103]={0.2, 0.4, 0.6 + 4. I}

```

1.7.2 matrices

<http://reference.wolfram.com/mathematica/tutorial/BasicMatrixOperations.html>

Matrices are lists of lists, where each sublist describes a row of the matrix:

```

1 In[104]:= M = {{3,2,7},{1,1,2},{0,-1,5},{2,2,1}};
2 In[105]:= Dimensions[M]
3 Out[105]={4, 3}

```

In this example, `M` is a 4×3 matrix. Pretty-printing a matrix is done with the `MatrixForm` wrapper,

```

1 In[106]:= MatrixForm[M]

```

Accessing matrix elements is analogous to accessing vector elements:

```

1 In[107]:= M[[1,3]]
2 Out[107]= 7
3 In[108]:= M[[2]]
4 Out[108]={1, 1, 2}

```

Matrices can be transposed with `Transpose[M]`.

Matrix–vector and matrix–matrix multiplications are done with the `.` operator:

```

1 In[109]:= M.a
2 Out[109]={2.8 + 14. I, 0.9 + 4. I, 1.3 + 10. I, 0.9 + 2. I}

```

1.7.3 sparse vectors and matrices

<http://reference.wolfram.com/mathematica/tutorial/SparseArrays-ManipulatingLists.html>

Large matrices can take up enormous amounts of computer memory. But in practical situations we are often dealing with matrices which are “sparse”, meaning that most of their entries are zero. A much more efficient way of storing them is therefore as a list of only their nonzero elements, using the `SparseArray` function.

A given vector or matrix is converted to sparse representation with

```

1 In[110]:= M = {{0,3,0,0,0,0,0,0,0},
2             {0,0,0,-1,0,0,0,0,0},
3             {0,0,0,0,0,0,0,0,0}};
4 In[111]:= Ms = SparseArray[M]
5 Out[111]= SparseArray[<2>, {3, 10}]

```

where the output shows that `Ms` is a 3×10 sparse matrix with 2 non-zero entries. We could have entered this matrix more easily by giving the list of non-zero entries,

```
1 In[112]:=Ms = SparseArray[{{1, 2} -> 3, {2, 4} -> -1}, {3, 10}];
```

which we can find out from

```
1 In[113]:=ArrayRules[Ms]
2 Out[113]={{1, 2} -> 3, {2, 4} -> -1, {_, _} -> 0}
```

which includes a specification of the default pattern `{_, _}`. This sparse array is converted back into a normal array with

```
1 In[114]:=Normal[Ms]
2 Out[114]={{0, 3, 0, 0, 0, 0, 0, 0, 0, 0},
3           {0, 0, 0, -1, 0, 0, 0, 0, 0, 0},
4           {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}
```

Sparse arrays and vectors can be used just like full arrays and vectors (they are internally converted automatically whenever necessary). But for some linear algebra operations they can be much more efficient. A matrix multiplication of two sparse matrices, for example, scales only with the number of non-zero elements of the matrices, not with their size.

1.7.4 matrix diagonalization

“Solving” the time-independent Schrödinger equation, as we will be doing in section 2.2, involves calculating the eigenvalues and eigenvectors of Hermitian³ matrices.

In what follows it is assumed that we have defined H as a Hermitian matrix. As an example we will use

```
1 In[115]:=H = {{0, 0.3, I, 0},
2             {0.3, 1, 0, 0},
3             {-I, 0, 1, -0.2},
4             {0, 0, -0.2, 3}};
```

eigenvalues

The eigenvalues of a matrix `H` are computed with

```
1 In[116]:=Eigenvalues[H]
2 Out[116]={3.0237, 1.63842, 0.998322, -0.660442}
```

Notice that these eigenvalues (energy values) are not necessarily sorted, even though in this example they appear in descending order. For a sorted list we use

³A complex matrix H is *Hermitian* if $H = H^\dagger$.


```

1 In[117]:=Sort[Eigenvalues[H]]
2 Out[117]={-0.660442, 0.998322, 1.63842, 3.0237}

```

For very large matrices H , and in particular for sparse matrices (see section 1.7.3), it is computationally inefficient to calculate all eigenvalues. Further, we are often only interested in the lowest-energy eigenvalues and eigenvectors. There are very efficient algorithms for calculating extremal eigenvalues,⁴ which can be used by specifying options to the `Eigenvalues` function: if we only need the largest two eigenvalue, for example, we call

```

1 In[118]:=Eigenvalues[H, 2, Method -> {"Arnoldi",
2                                     "Criteria" -> "RealPart",
3                                     MaxIterations -> 10^6}]
4 Out[118]={3.0237, 1.63842}

```

Unfortunately there is no direct way to calculate the *smallest* eigenvalues; but since the smallest eigenvalues of H are the largest eigenvalues of $-H$ we can use

```

1 In[119]:=-Eigenvalues[-H, 2, Method -> {"Arnoldi",
2                                     "Criteria" -> "RealPart",
3                                     MaxIterations -> 10^6}]
4 Out[119]={0.998322, -0.660442}

```

eigenvectors

The eigenvectors of a matrix H are computed with

```

1 In[120]:=Eigenvectors[H]
2 Out[120]={ {0.-0.0394613I, 0.-0.00584989I, -0.117564, 0.992264},
3           {0.+0.533642I, 0.+0.250762I, 0.799103, 0.117379},
4           {0.-0.0053472I, 0.+0.955923I, -0.292115, -0.029187},
5           {0.-0.844772I, 0.+0.152629I, 0.512134, 0.0279821}}

```

In this case of a 4×4 matrix, this generates a list of four 4-vectors which are orthogonal.

Usually we are interested in calculating the eigenvalues and eigenvectors at the same time:

```

1 In[121]:=Eigensystem[H]
2 Out[121]={ {3.0237, 1.63842, 0.998322, -0.660442},
3           { {0.-0.0394613I, 0.-0.00584989I, -0.117564, 0.992264},
4             {0.+0.533642I, 0.+0.250762I, 0.799103, 0.117379},
5             {0.-0.0053472I, 0.+0.955923I, -0.292115, -0.029187},
6             {0.-0.844772I, 0.+0.152629I, 0.512134, 0.0279821}} }

```

⁴Lanczos algorithm: http://en.wikipedia.org/wiki/Lanczos_algorithm

which generates a list containing the eigenvalues and the eigenvectors. The ordering of the elements in the eigenvalues list corresponds to the ordering in the eigenvectors list; but the sorting order is generally undefined. To generate a list of (eigenvalue, eigenvector) pairs in ascending order of eigenvalues, we calculate

```

1 In[122]:=Sort[Transpose[Eigensystem[H]]]
2 Out[122]={{-0.660442,
3           {0.-0.844772I, 0.+0.152629I, 0.512134, 0.0279821}},
4           {0.998322,
5           {0.-0.0053472I, 0.+0.955923I, -0.292115, -0.029187}},
6           {1.63842,
7           {0.+0.533642I, 0.+0.250762I, 0.799103, 0.117379}},
8           {3.0237,
9           {0.-0.0394613I, 0.-0.00584989I, -0.117564, 0.992264}}}
```

To generate a sorted list of eigenvalues `eval` and a corresponding list of eigenvectors `evect` we calculate

```

1 In[123]:={eval, evect} = Transpose[Sort[Transpose[Eigensystem[H]]]];
2 In[124]:=eval
3 Out[124]={-0.660442, 0.998322, 1.63842, 3.0237}
4 In[125]:=evect
5 Out[125]={0.-0.844772I, 0.+0.152629I, 0.512134, 0.0279821},
6           {0.-0.0053472I, 0.+0.955923I, -0.292115, -0.029187},
7           {0.+0.533642I, 0.+0.250762I, 0.799103, 0.117379},
8           {0.-0.0394613I, 0.-0.00584989I, -0.117564, 0.992264}}
```

The trick with calculating only the lowest-energy eigenvalues can be applied to eigenvalue calculations as well, since the eigenvectors of $-H$ and H are the same:

```

1 In[126]:={eval, evect} = Transpose[Sort[Transpose[-Eigensystem[-H, 2,
2           Method -> {"Arnoldi", "Criteria" -> "RealPart",
3           MaxIterations -> 10^6}]]]];
4 In[127]:=eval
5 Out[127]={-0.660442, 0.998322}
6 In[128]:=evect
7 Out[128]={{-0.733656+0.418794I, 0.132553-0.0756656I,
8           -0.253889-0.444771I, -0.0138721-0.0243015 I},
9           {-0.000575666-0.00531612I, 0.102912+0.950367I,
10          -0.290417+0.0314484I, -0.0290174+0.0031422I}}
```

Notice that these eigenvectors are not the same as those calculated further above! This difference is due to arbitrary multiplications of the eigenvectors with phase factors $e^{i\varphi}$.

To check that the vectors in `evect` are ortho-normalized, we calculate the matrix product

```

1 In[129]:=Conjugate[evect].Transpose[evect] // Chop // MatrixForm
```

and verify that the matrix of scalar products is indeed equal to the unit matrix.

To check that the vectors in `evvec` are indeed eigenvectors of `H`, we calculate all matrix elements of `H` in this basis of eigenvectors:

```
1 In[130]:=Conjugate[evvec].H.Transpose[evvec] // Chop // MatrixForm
```

and verify that the result is a diagonal matrix whose diagonal elements are exactly the eigenvalues `eval`.

1.7.5 exercises

Q1.19 Calculate the eigenvalues and eigenvectors of the Pauli matrices:

http://en.wikipedia.org/wiki/Pauli_matrices

Are the eigenvectors ortho-normal? If not, find an ortho-normal set.

1.8 complex numbers

By default all variables in Mathematica are assumed to be complex numbers, unless otherwise specified. All mathematical functions can take complex numbers as their input, often by analytic continuation.

The most commonly used functions on complex numbers are `Conjugate`, `Re`, `Im`, `Abs`, and `Arg`. When applied to numerical arguments they do what we expect:

```
1 In[131]:=Conjugate[2 + 3*I]
2 Out[131]=2 - 3*I
3 In[132]:=Im[0.7]
4 Out[132]=0
```

When applied to variable arguments, however, they fail and frustrate the inexperienced user:

```
1 In[133]:=Conjugate[x+I*y]
2 Out[133]=Conjugate[x] - I*Conjugate[y]
3 In[134]:=Im[a]
4 Out[134]=Im[a]
```

This behavior is due to Mathematica not knowing that `x`, `y`, and `a` in these examples are real-valued. There are several ways around this, all involving *assumptions*. The first is to use the `ComplexExpand` function, which assumes that all variables are *real*:

```
1 In[135]:=Conjugate[x+I*y] // ComplexExpand
2 Out[135]=x - I*y
3 In[136]:=Im[a] // ComplexExpand
4 Out[136]=0
```

The second is to use explicit *local* assumptions, which may be more specific than assuming that all variables are real-valued:

```
1 In[137]:=Assuming[Element[x, Reals] && Element[y, Reals],
2 Conjugate[x + I y] // FullSimplify]
```

```

3 Out[137]=x - I*y
4 In[138]:=Assuming[Element[a, Reals], Im[a]]
5 Out[138]=0

```

The third is to use *global* assumptions (in general, global system variables start with the \$ sign):

```

1 In[139]:=$Assumptions = Element[x, Reals] && Element[y, Reals] &&
2           Element[a, Reals];
3 In[140]:=Conjugate[x+I*y] // FullSimplify
4 Out[140]=x - I*y
5 In[141]:=Im[a] // FullSimplify
6 Out[141]=0

```

1.9 units

<http://reference.wolfram.com/mathematica/tutorial/UnitsOverview.html>

Mathematica is capable of dealing with units of measure, as required for physical calculations. For example, we can make the assignment

```

1 In[142]:=s = Quantity["3 m"];

```

to specify that *s* should be three meters. A large number of units can be used, as well as physical constants:

```

1 In[143]:=kB = Quantity["BoltzmannConstant"];

```

will define the variable *kB* to be Boltzmann's constant. Take note that complicated or slightly unusual quantities are evaluated through the online service *Wolfram Alpha*, which means that you need an internet connection in order to evaluate them.

In principle, we can use this mechanism to do all the calculations in this lecture with units; however, for the sake of generality (as many other computer programs cannot deal with units) when we do numerical calculations, we will convert every quantity into dimensionless form in what follows.

In order to eliminate units from a calculation, we must determine a set of units in which to express the relevant quantities. This means that every physical quantity *x* is expressed as the product of a *unit* x_0 and a *dimensionless multiplier* x' . The actual calculations are performed only with the dimensionless multipliers. For example, a length $s = 3 \text{ m}$ can be expressed with the unit $s_0 = 1 \text{ m}$ and $s' = 3$, such that $s's_0 = s$. It can equally well be expressed with $s_0 = 52.9177 \text{ pm}$ (the Bohr radius) and $s' = 5.66918 \times 10^{10}$. A smart choice of units can help in implementing a problem.

As an example we calculate the acceleration of an A380 airplane ($m = 560 \text{ t}$) due to its jet engines ($F = 4 \times 311 \text{ kN}$). The easiest way is to use Mathematica's built-in unit processing:

```

1 In[144]:=F = Quantity["4*311 kN"];
2 In[145]:=m = Quantity["560 t"];
3 In[146]:=a = UnitConvert[F/m, "m/s^2"] //N
4 Out[146]=2.22143 m/s^2

```

Now we do the same calculation without Mathematica's unit processing. Setting $F = F' F_0$, $m = m' m_0$, and $a = a' a_0$, Newton's equation $F = ma$ can be solved for the dimensionless acceleration a' :

$$a' = \frac{F'}{m'} \times \frac{F_0}{m_0 a_0}. \quad (1.1)$$

SI units: With SI units ($F_0 = 1 \text{ N}$, $m_0 = 1 \text{ kg}$, $a_0 = 1 \text{ m/s}^2$), the last term of Eq. (1.1) is $F_0/(m_0 a_0) = 1$, which simplifies the calculation greatly. This is the advantage of SI units.

With $F' = 1244000$ and $m' = 560000$ we find $a' = F'/m' = 2.22143$. Therefore we know that the airplane's acceleration will be $a = a' a_0 = 2.22143 \text{ m/s}^2$.

Arbitrary units: If we choose, for example, the units

- $F_0 = 1000 \text{ N}$, the maximum force a human can apply, as the unit of force,
- $m_0 = 5 \text{ g}$, the weight of a hummingbird, as the unit of mass,
- $a_0 = 9.81 \text{ m/s}^2$, the earth's gravitational acceleration, as the unit of acceleration,

the last term $k = F_0/(m_0 a_0)$ of Eq. (1.1) is computed in Mathematica with

```
1 In[147]:=F0 = Quantity["1000 N"];
2 In[148]:=m0 = Quantity["5 g"];
3 In[149]:=a0 = Quantity["9.81 m/s^2"];
4 In[150]:=k = F0/(m0 a0)
5 Out[150]=20387.4
```

and we find the airplane's acceleration with

```
1 In[151]:=F = Quantity["4*311 kN"]/F0 //N
2 Out[151]=1244.
3 In[152]:=m = Quantity["560 t"]/m0 //N
4 Out[152]=1.12*10^8
5 In[153]:=a = F/m * k
6 Out[153]=0.226445
```

Thus we know that the acceleration is $0.226445g$, which is

```
1 In[154]:=a*a0
2 Out[154]=2.22143 m/s^2
```


Chapter 2

quantum mechanics

In this chapter we connect quantum mechanics to representations that a computer can understand.

2.1 basis sets and representations

Quantum-mechanical problems are usually specified in terms of operators and wavefunctions. The wavefunctions are elements of a Hilbert space; the operators act on such vectors. How can these objects be represented on a computer, which only understands numbers but not Hilbert spaces?

In order to find a computer-representable form of these abstract objects, we assume that we know an ortho-normal¹ basis $\{|i\rangle\}_i$ of this Hilbert space, with scalar product $\langle i|j\rangle = \delta_{ij}$. In section 2.4 we will talk about how to construct such bases. For now we make the assumption that this basis is complete, such that $\sum_i |i\rangle\langle i| = \mathbb{1}$. We will see in section 2.1.1 how to deal with incomplete basis sets.

Given any operator $\hat{\mathcal{A}}$ acting on this Hilbert space, we use the completeness relation twice to find

$$\hat{\mathcal{A}} = \mathbb{1} \cdot \hat{\mathcal{A}} \cdot \mathbb{1} = \left[\sum_i |i\rangle\langle i| \right] \cdot \hat{\mathcal{A}} \cdot \left[\sum_j |j\rangle\langle j| \right] = \sum_{ij} \langle i|\hat{\mathcal{A}}|j\rangle |i\rangle\langle j|. \quad (2.1)$$

If we define a numerical matrix \mathbf{A} with elements $A_{ij} = \langle i|\hat{\mathcal{A}}|j\rangle \in \mathbb{C}$ we rewrite this as

$$\hat{\mathcal{A}} = \sum_{ij} A_{ij} |i\rangle\langle j|. \quad (2.2)$$

The same can be done with a state vector $|\psi\rangle$: using the completeness relation,

$$|\psi\rangle = \mathbb{1} \cdot |\psi\rangle = \left[\sum_i |i\rangle\langle i| \right] \cdot |\psi\rangle = \sum_i \langle i|\psi\rangle |i\rangle, \quad (2.3)$$

and defining a numerical vector $\vec{\psi}$ with elements $\psi_i = \langle i|\psi\rangle \in \mathbb{C}$ the state vector is

$$|\psi\rangle = \sum_i \psi_i |i\rangle. \quad (2.4)$$

¹The following calculations can be extended to situations where the basis is not ortho-normal. For the scope of this lecture we are however not interested in this complication.

Both the matrix A and the vector $\tilde{\psi}$ are complex-valued objects which can be represented in any computer system. Equations (2.2) and (2.4) serve to convert between Hilbert-space representations and number-based (matrix/vector-based) representations. These equations are at the center of what it means to find a computer representation of a quantum-mechanical problem.

2.1.1 incomplete basis sets

For infinite-dimensional Hilbert spaces we must usually content ourselves with finite basis sets which approximate the low-energy physics (or, more generally, the physically relevant dynamics) of the problem. In practice this means that an orthonormal basis set may not be complete:

$$\sum_i |i\rangle\langle i| = \hat{P} \quad (2.5)$$

which is the projector onto that subspace of the full Hilbert space that the basis is capable of describing. We denote $\hat{Q} = \mathbb{1} - \hat{P}$ as the complement of this projector: \hat{Q} is the projector onto the remainder of the Hilbert space that is left out of this truncated description. The equivalent of Eq. (2.1) is then

$$\begin{aligned} \hat{\mathcal{A}} &= \mathbb{1} \cdot \hat{\mathcal{A}} \cdot \mathbb{1} = (\hat{P} + \hat{Q}) \cdot \hat{\mathcal{A}} \cdot (\hat{P} + \hat{Q}) = \hat{P} \cdot \hat{\mathcal{A}} \cdot \hat{P} + \hat{P} \cdot \hat{\mathcal{A}} \cdot \hat{Q} + \hat{Q} \cdot \hat{\mathcal{A}} \cdot \hat{P} + \hat{Q} \cdot \hat{\mathcal{A}} \cdot \hat{Q} \\ &= \underbrace{\sum_{ij} A_{ij} |i\rangle\langle j|}_{\text{within described subspace}} + \underbrace{\hat{P} \cdot \hat{\mathcal{A}} \cdot \hat{Q} + \hat{Q} \cdot \hat{\mathcal{A}} \cdot \hat{P}}_{\text{neglected coupling to (high-energy) part}} + \underbrace{\hat{Q} \cdot \hat{\mathcal{A}} \cdot \hat{Q}}_{\text{neglected (high-energy) part}} \end{aligned} \quad (2.6)$$

In the same way, the equivalent of Eq. (2.3) is

$$|\psi\rangle = \mathbb{1} \cdot |\psi\rangle = (\hat{P} + \hat{Q}) \cdot |\psi\rangle = \underbrace{\sum_i \psi_i |i\rangle}_{\text{within described subspace}} + \underbrace{\hat{Q}|\psi\rangle}_{\text{neglected (high-energy) part}} \quad (2.7)$$

Since \hat{Q} is the projector onto the neglected subspace, the component $\hat{Q}|\psi\rangle$ of Eq. (2.7) is the part of the wavefunction $|\psi\rangle$ that is left out of the description in the truncated basis. In specific situations we will need to make sure that all terms involving \hat{Q} in Eqs. (2.6) and (2.7) can be safely neglected.

2.1.2 exercises

Q2.1 We describe a spin-1/2 system in the basis containing the two states

$$\begin{aligned} |\uparrow(\vartheta, \varphi)\rangle &= \cos\left(\frac{\vartheta}{2}\right) |\uparrow\rangle + e^{i\varphi} \sin\left(\frac{\vartheta}{2}\right) |\downarrow\rangle \\ |\downarrow(\vartheta, \varphi)\rangle &= -e^{-i\varphi} \sin\left(\frac{\vartheta}{2}\right) |\uparrow\rangle + \cos\left(\frac{\vartheta}{2}\right) |\downarrow\rangle \end{aligned} \quad (2.8)$$

1. Show that this basis is orthonormal.
2. Express the states $|\uparrow\rangle$ and $|\downarrow\rangle$ as vectors in this basis.
3. Express the Pauli operators as matrices in this basis.

4. Show that $|\uparrow(\vartheta, \varphi)\rangle$ and $|\downarrow(\vartheta, \varphi)\rangle$ are eigenvectors of $\hat{\sigma}(\vartheta, \varphi) = \hat{\sigma}_x \sin \vartheta \cos \varphi + \hat{\sigma}_y \sin \vartheta \sin \varphi + \hat{\sigma}_z \cos \vartheta$. What are the eigenvalues?

Q2.2 The eigenstate basis for the description of the infinite square well of unit width is made up of the ortho-normalized functions

$$\langle x|n\rangle = \phi_n(x) = \sqrt{2} \sin(n\pi x) \quad (2.9)$$

defined on the interval $[0, 1]$, with $n \in \{1, 2, 3, \dots\}$.

1. Calculate the function $P_\infty(x, y) = \langle x| \left[\sum_{n=1}^{\infty} |n\rangle \langle n| \right] |y\rangle$.
2. In computer-based calculations we limit the basis set to $n \in \{1, 2, 3, \dots, n_{\max}\}$ for some large value of n_{\max} . Using Mathematica, calculate the function $P_{n_{\max}}(x, y) = \langle x| \left[\sum_{n=1}^{n_{\max}} |n\rangle \langle n| \right] |y\rangle$ (use the [Sum](#) function). Make a plot for $n_{\max} = 100$ (use the [DensityPlot](#) function).
3. What does the function P represent?

2.2 time-independent Schrödinger equation

The time-independent Schrödinger equation is

$$\hat{\mathcal{H}}|\psi\rangle = E|\psi\rangle. \quad (2.10)$$

As in section 2.1 we use a computational basis to express the Hamiltonian operator $\hat{\mathcal{H}}$ and the wavefunction ψ as

$$\begin{aligned} \hat{\mathcal{H}} &= \sum_{ij} H_{ij} |i\rangle \langle j| \\ |\psi\rangle &= \sum_i \psi_i |i\rangle \end{aligned} \quad (2.11)$$

With these substitutions the Schrödinger equation becomes

$$\begin{aligned} \left[\sum_{ij} H_{ij} |i\rangle \langle j| \right] \left[\sum_k \psi_k |k\rangle \right] &= E \left[\sum_i \psi_i |i\rangle \right] \\ \sum_{ijk} H_{ij} \psi_k \langle j|k\rangle |i\rangle &= \sum_i E \psi_i |i\rangle \\ \sum_{ij} H_{ij} \psi_j |i\rangle &= \sum_i E \psi_i |i\rangle \end{aligned} \quad (2.12)$$

Multiplying this equation by $\langle \ell|$ from the left, and using the orthonormality of the basis set, gives

$$\begin{aligned} \sum_{ij} H_{ij} \psi_j \langle \ell|i\rangle &= \sum_i E \psi_i \langle \ell|i\rangle \\ \sum_j H_{\ell j} \psi_j &= E \psi_\ell \end{aligned} \quad (2.13)$$

In matrix notation this can be written as

$$\boxed{H \cdot \vec{\psi} = E \vec{\psi}.} \quad (2.14)$$

This is the central equation of this lecture. It is the time-independent Schrödinger equation in a form that computers can understand, namely an eigenvalue equation in terms of numerical (complex) matrices and vectors.

If you think that there is no difference between Eqs. (2.10) and (2.14), then I invite you to re-read this section as I consider it extremely important for what follows in this course. You can think of Eq. (2.10) as an abstract relationship between operators and vectors in Hilbert space, while Eq. (2.14) is a *numerical representation* of this relationship in a concrete basis set $\{|i\rangle\}_i$. They both contain the exact same information (since we converted one to the other in a few lines of mathematics) but they are conceptually very different, as one is understandable by a computer and the other is not.

2.2.1 diagonalization

The matrix form (2.14) of the Schrödinger equation is an eigenvalue equation as you know from linear algebra. Given a matrix of complex numbers \mathbf{H} we can find the eigenvalues E_i and eigenvectors $\tilde{\psi}_i$ using Mathematica's built-in procedures, as described in section 1.7.4.

2.2.2 exercises

Q2.3 Express the spin-1/2 Hamiltonian

$$\hat{\mathcal{H}} = \sin(\vartheta) \cos(\varphi) \hat{S}_x + \sin(\vartheta) \sin(\varphi) \hat{S}_y + \cos(\vartheta) \hat{S}_z \quad (2.15)$$

in the basis $\{|\uparrow\rangle, |\downarrow\rangle\}$, and calculate its eigenvalues and eigenvectors.

2.3 time-dependent Schrödinger equation

The time-dependent Schrödinger equation is

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = \hat{\mathcal{H}}(t) |\psi(t)\rangle, \quad (2.16)$$

where the Hamiltonian $\hat{\mathcal{H}}$ can have an explicit time dependence. This differential equation has the formal solution

$$|\psi(t)\rangle = \hat{\mathcal{U}}(t_0; t) |\psi(t_0)\rangle \quad (2.17)$$

in terms of the *propagator*

$$\begin{aligned} \hat{\mathcal{U}}(t_0; t) = & \mathbb{1} - \frac{i}{\hbar} \int_{t_0}^t dt_1 \hat{\mathcal{H}}(t_1) - \frac{1}{\hbar^2} \int_{t_0}^t dt_1 \int_{t_0}^{t_1} dt_2 \hat{\mathcal{H}}(t_1) \hat{\mathcal{H}}(t_2) \\ & + \frac{i}{\hbar^3} \int_{t_0}^t dt_1 \int_{t_0}^{t_1} dt_2 \int_{t_0}^{t_2} dt_3 \hat{\mathcal{H}}(t_1) \hat{\mathcal{H}}(t_2) \hat{\mathcal{H}}(t_3) \\ & + \frac{1}{\hbar^4} \int_{t_0}^t dt_1 \int_{t_0}^{t_1} dt_2 \int_{t_0}^{t_2} dt_3 \int_{t_0}^{t_3} dt_4 \hat{\mathcal{H}}(t_1) \hat{\mathcal{H}}(t_2) \hat{\mathcal{H}}(t_3) \hat{\mathcal{H}}(t_4) + \dots \end{aligned} \quad (2.18)$$

which propagates any state from time t_0 to time t . An alternative form is given by the *Magnus expansion*²

$$\hat{\mathcal{U}}(t_0; t) = \exp \left[\sum_{k=1}^{\infty} \hat{\Omega}_k(t_0; t) \right] \quad (2.19)$$

with the contributions

$$\begin{aligned} \hat{\Omega}_1(t_0; t) &= -\frac{i}{\hbar} \int_{t_0}^t dt_1 \hat{\mathcal{H}}(t_1) \\ \hat{\Omega}_2(t_0; t) &= -\frac{1}{2\hbar^2} \int_{t_0}^t dt_1 \int_{t_0}^{t_1} dt_2 [\hat{\mathcal{H}}(t_1), \hat{\mathcal{H}}(t_2)] \\ \hat{\Omega}_3(t_0; t) &= \frac{i}{6\hbar^3} \int_{t_0}^t dt_1 \int_{t_0}^{t_1} dt_2 \int_{t_0}^{t_2} dt_3 ([\hat{\mathcal{H}}(t_1), [\hat{\mathcal{H}}(t_2), \hat{\mathcal{H}}(t_3)]] + [\hat{\mathcal{H}}(t_3), [\hat{\mathcal{H}}(t_2), \hat{\mathcal{H}}(t_1)]]) \\ &\dots \end{aligned} \quad (2.20)$$

This expansion in terms of different-time commutators is often easier to evaluate than Eq. (2.18), especially when the contributions vanish for $k > k_{\max}$ (see section 2.3.3 for the case $k_{\max} = 1$). Even if higher-order contributions do not vanish entirely, they (usually) decrease in importance much more rapidly with increasing k than those of Eq. (2.18). Also, even if the Magnus expansion is artificially truncated (neglecting higher-order terms), the quantum-mechanical evolution is still unitary; this is not the case for Eq. (2.18).

2.3.1 time-independent basis

We again express the wavefunction in terms of the chosen basis, which is assumed to be time-independent. This leaves the time dependence in the expansion coefficients,

$$\begin{aligned} \hat{\mathcal{H}}(t) &= \sum_{ij} H_{ij}(t) |i\rangle\langle j| \\ |\psi(t)\rangle &= \sum_i \psi_i(t) |i\rangle. \end{aligned} \quad (2.21)$$

Inserting these expressions into the time-dependent Schrödinger equation (2.16) gives

$$i\hbar \sum_i \dot{\psi}_i(t) |i\rangle = \left[\sum_{ij} H_{ij}(t) |i\rangle\langle j| \right] \sum_k \psi_k(t) |k\rangle = \sum_{ij} H_{ij}(t) \psi_j(t) |i\rangle. \quad (2.22)$$

Multiplying with $\langle \ell |$ from the left:

$$i\hbar \dot{\psi}_\ell(t) = \sum_j H_{\ell j}(t) \psi_j(t) \quad (2.23)$$

or, in matrix notation,

$$i\hbar \dot{\vec{\psi}}(t) = \mathbf{H}(t) \cdot \vec{\psi}(t). \quad (2.24)$$

Since the matrix $\mathbf{H}(t)$ is supposedly known, this equation represents a system of coupled complex differential equations for the vector $\vec{\psi}(t)$, which can be solved on a computer.

²http://en.wikipedia.org/wiki/Magnus_expansion

2.3.2 time-dependent basis: interaction picture

It can be advantageous to use a time-dependent basis. The most frequently used such basis is given by the interaction picture of quantum mechanics, where the Hamiltonian can be split into a time-independent principal part $\hat{\mathcal{H}}_0$ and a small time-dependent part $\hat{\mathcal{H}}_1$:

$$\hat{\mathcal{H}}(t) = \hat{\mathcal{H}}_0 + \hat{\mathcal{H}}_1(t). \quad (2.25)$$

Assuming that we can diagonalize $\hat{\mathcal{H}}_0$, possibly numerically, such that the eigenfunctions satisfy $\hat{\mathcal{H}}_0|i\rangle = E_i|i\rangle$, we propose the time-dependent basis

$$|i(t)\rangle = e^{-iE_i t/\hbar}|i\rangle. \quad (2.26)$$

If we express any wavefunction in this basis as

$$|\psi(t)\rangle = \sum_i \psi_i(t) |i(t)\rangle = \sum_i \psi_i(t) e^{-iE_i t/\hbar} |i\rangle, \quad (2.27)$$

the time-dependent Schrödinger equation becomes

$$\begin{aligned} \sum_i [\hbar i \dot{\psi}_i(t) + E_i \psi_i(t)] e^{-iE_i t/\hbar} |i\rangle &= \sum_i \psi_i(t) e^{-iE_i t/\hbar} E_i |i\rangle + \sum_i \psi_i(t) e^{-iE_i t/\hbar} \hat{\mathcal{H}}_1(t) |i\rangle \\ \hbar i \dot{\psi}_i(t) &= \sum_j \psi_j(t) e^{-i(E_j - E_i)t/\hbar} \langle i | \hat{\mathcal{H}}_1(t) | j \rangle. \end{aligned} \quad (2.28)$$

This is the same matrix/vector evolution expression as Eq. (2.24), except that here the Hamiltonian matrix elements must be defined as

$$H_{ij}(t) = \langle i | \hat{\mathcal{H}}_1(t) | j \rangle e^{-i(E_j - E_i)t/\hbar}. \quad (2.29)$$

We see immediately that if the interaction Hamiltonian vanishes [$\hat{\mathcal{H}}_1(t) = 0$], then the expansion coefficients $\psi_i(t)$ become time-independent, as expected since they are the coefficients of the eigenfunctions of the time-independent Schrödinger equation.

When a quantum-mechanical system is composed of different parts which have vastly different energy scales of their internal evolution $\hat{\mathcal{H}}_0$, then the use of Eq. (2.29) can have great numerical advantages. It turns out that the relevant interaction terms $H_{ij}(t)$ in the interaction picture will have relatively slowly evolving phases $\exp[-i(E_j - E_i)t/\hbar]$, on a time scale given by relative energy *differences* and not by *absolute* energies; this makes it possible to numerically solve the coupled differential equations of Eq. (2.24) without using an absurdly small time step.

2.3.3 special case: $[\hat{\mathcal{H}}(t), \hat{\mathcal{H}}(t')] = 0 \forall (t, t')$

If the Hamiltonian commutes with itself at different times, $[\hat{\mathcal{H}}(t), \hat{\mathcal{H}}(t')] = 0 \forall (t, t')$, the propagator (2.19) of Eq. (2.16) can be simplified to

$$\hat{\mathcal{U}}(t_0; t) = \exp \left[-\frac{i}{\hbar} \int_{t_0}^t \hat{\mathcal{H}}(s) ds \right], \quad (2.30)$$

and the corresponding solution of Eq. (2.24) is

$$\vec{\psi}(t) = \exp \left[-\frac{i}{\hbar} \int_{t_0}^t \mathbf{H}(s) ds \right] \cdot \vec{\psi}(t_0). \quad (2.31)$$

Notice that exponential in this expression has a matrix as its argument: in Mathematica this matrix exponentiation is done with the [MatrixExp](#) function.

2.3.4 special case: time-independent Hamiltonian

In the special (but common) case where the Hamiltonian is time-independent, the integral in Eq. (2.31) can be evaluated immediately, and the solution is

$$\vec{\psi}(t) = \exp\left[-\frac{i(t-t_0)}{\hbar}\mathbf{H}\right] \cdot \vec{\psi}(t_0). \quad (2.32)$$

If we have a specific Hamiltonian matrix \mathbf{H} defined, for example the matrix of section 1.7.4, we can calculate the propagator $\mathbf{U}(t) = \exp(-i\mathbf{H}t/\hbar)$ with

```
1 In[155]:=U[t_] = MatrixExp[-I H t]
```

where we have used $\mathbf{t} = (t - t_0)/\hbar$ by expressing time in units of inverse energy (see section 1.9). The resulting expression for $\mathbf{U}[\mathbf{t}]$ will in general be very long.

2.3.5 exercises

Q2.4 Demonstrate that the propagator of Eq. (2.30) gives a wavefunction (2.17) which satisfies Eq. (2.16).

Q2.5 Calculate the propagator of the Hamiltonian of **Q2.3** (page 34).

2.4 basis construction

In principle, the choice of basis set $\{|i\rangle\}_i$ does not influence the way a computer program like Mathematica solves a quantum-mechanical problem. In practice, however, we always need a *constructive* way to find some basis for a given quantum-mechanical problem. A basis which takes the system's Hamiltonian into account may give a computationally simpler description; however, in complicated systems it is often more important to find *any* way of constructing a usable basis set that finding the perfect one.

2.4.1 description of a single degree of freedom

When we describe a single quantum-mechanical degree of freedom, it is often possible to deduce a useful basis set from knowledge of the Hilbert space itself. This is what we will be doing in chapter 3 for spin systems, where the well-known Dicke basis $\{|S, M_S\rangle\}_{M_S=-S}^S$ turns out to be very useful.

For more complicated degrees of freedom, we can find inspiration for a basis choice from an associated Hamiltonian. Such Hamiltonians describing a single degree of freedom are often so simple that they can be diagonalized by hand. If this is not the case, real-world Hamiltonians $\hat{\mathcal{H}}$ can often be decomposed into a “simple” part $\hat{\mathcal{H}}_0$ that is time-independent and can be diagonalized easily, and a “difficult” part $\hat{\mathcal{H}}_1$ that usually contains complicated interactions and/or time-dependent terms but is of smaller magnitude:

$$\hat{\mathcal{H}}(t) = \hat{\mathcal{H}}_0 + \hat{\mathcal{H}}_1(t). \quad (2.33)$$

A natural choice of basis set is the set of eigenstates of $\hat{\mathcal{H}}_0$, or at least those eigenstates below a certain cutoff energy since they will be optimally suited to describe

the complete low-energy behavior of the degree of freedom in question. This latter point is especially important for infinite-dimensional systems (chapter 4), where any computer representation will necessarily truncate the dimensionality, as discussed in section 2.1.1.

examples of basis sets for single degrees of freedom:

spin degree of freedom: Dicke states $|S, M_S\rangle$

translational degree of freedom: square-well eigenstates, harmonic oscillator eigenstates

rotational degree of freedom: spherical harmonics

atomic system: hydrogen-like orbitals

translation-invariant system: periodic plane waves (reciprocal lattice)

2.4.2 description of coupled degrees of freedom

A broad range of quantum-mechanical systems of interest are governed by Hamiltonians of the form

$$\hat{\mathcal{H}} = \sum_{k=1}^N \hat{\mathcal{H}}^{(k)} + \hat{\mathcal{H}}_{\text{int}}(t), \quad (2.34)$$

where N individual degrees of freedom are governed by their individual Hamiltonians $\hat{\mathcal{H}}^{(k)}$, while their interactions are described by $\hat{\mathcal{H}}_{\text{int}}$. This is a situation we will encounter repeatedly as we construct more complicated quantum-mechanical problems from simpler parts. A few simple examples are:

- A set of N interacting particles: the Hamiltonians $\hat{\mathcal{H}}^{(k)}$ describe the individual particles, while $\hat{\mathcal{H}}_{\text{int}}$ describes their interactions.
- A single particle moving in three spatial degrees of freedom: the Hamiltonians $\hat{\mathcal{H}}^{(x,y,z)}$ describe the kinetic energy in the three directions, while $\hat{\mathcal{H}}_{\text{int}}$ contains the potential energy.
- A single particle with internal (spin) and external (motional) degrees of freedom which are coupled through a state-dependent potential in $\hat{\mathcal{H}}_{\text{int}}$.

The existence of individual Hamiltonians $\hat{\mathcal{H}}^{(k)}$ assumes that the Hilbert space of the complete system has a tensor-product structure

$$V = V^{(1)} \otimes V^{(2)} \otimes \dots \otimes V^{(N)}, \quad (2.35)$$

where each Hamiltonian $\hat{\mathcal{H}}^{(k)}$ acts only in a single component space,

$$\hat{\mathcal{H}}^{(k)} = \mathbb{1}^{(1)} \otimes \mathbb{1}^{(2)} \otimes \dots \otimes \mathbb{1}^{(k-1)} \otimes \hat{h}^{(k)} \otimes \mathbb{1}^{(k+1)} \otimes \dots \otimes \mathbb{1}^{(N)}. \quad (2.36)$$

Further, if we are able to construct bases $\{|i\rangle^{(k)}\}_{i=1}^{n_k}$ for all of the component Hilbert spaces $V^{(k)}$, as in section 2.4.1, then we can construct a basis for the full Hilbert space V by taking all possible tensor products of basis functions:

$$|i_1, i_2, \dots, i_N\rangle = |i_1\rangle^{(1)} \otimes |i_2\rangle^{(2)} \otimes \dots \otimes |i_N\rangle^{(N)}. \quad (2.37)$$

This basis will have $\prod_{k=1}^N n_k$ elements, which can easily become a very large number for composite systems.

wave vectors (quantum states)

A product state of the complete system

$$|\psi\rangle = |\psi\rangle^{(1)} \otimes |\psi\rangle^{(2)} \otimes \dots \otimes |\psi\rangle^{(N)} \quad (2.38)$$

can be described in the following way. First, each single-particle wavefunction is decomposed in its own basis as in Eq. (2.4),

$$|\psi\rangle^{(k)} = \sum_{i_k=1}^{n_k} \psi_{i_k}^{(k)} |i_k\rangle^{(k)}. \quad (2.39)$$

Inserting these expansions into Eq. (2.38) gives the expansion into the basis functions (2.37) of the full system,

$$\begin{aligned} |\psi\rangle &= \left[\sum_{i_1=1}^{n_1} \psi_{i_1}^{(1)} |i_1\rangle^{(1)} \right] \otimes \left[\sum_{i_2=1}^{n_2} \psi_{i_2}^{(2)} |i_2\rangle^{(2)} \right] \otimes \dots \otimes \left[\sum_{i_N=1}^{n_N} \psi_{i_N}^{(N)} |i_N\rangle^{(N)} \right] \\ &= \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \dots \sum_{i_N=1}^{n_N} \left[\psi_{i_1}^{(1)} \psi_{i_2}^{(2)} \dots \psi_{i_N}^{(N)} \right] |i_1, i_2, \dots, i_N\rangle \end{aligned} \quad (2.40)$$

In Mathematica, such a wavefunction tensor product can be calculated as follows. For example, assume that `psi1` is a vector containing the expansion of $|\psi\rangle^{(1)}$ in its basis, and similarly for `psi2` and `psi3`. The vector `psi` of expansion coefficients of the full wavefunction $|\psi\rangle = |\psi\rangle^{(1)} \otimes |\psi\rangle^{(2)} \otimes |\psi\rangle^{(3)}$ is calculated with

```
In[156]:=psi = Flatten[KroneckerProduct[psi1, psi2, psi3]]
```

See Eq. (2.44) for a numerical example as an exercise.

operators

If the Hilbert space has the tensor-product structure of Eq. (2.35), then the operators acting on this full space are often given as tensor products as well,

$$\hat{A} = \hat{a}^{(1)} \otimes \hat{a}^{(2)} \otimes \dots \otimes \hat{a}^{(N)}, \quad (2.41)$$

or as a sum over such products. If every single-particle operator is decomposed in its own basis as in Eq. (2.2),

$$\hat{a}^{(k)} = \sum_{i_k=1}^{n_k} \sum_{j_k=1}^{n_k} a_{i_k, j_k}^{(k)} |i_k\rangle^{(k)} \langle j_k|^{(k)}, \quad (2.42)$$

inserting these expressions into Eq. (2.41) gives the expansion into the basis functions (2.37) of the full system,

$$\begin{aligned} \hat{A} &= \left[\sum_{i_1=1}^{n_1} \sum_{j_1=1}^{n_1} a_{i_1, j_1}^{(1)} |i_1\rangle^{(1)} \langle j_1|^{(1)} \right] \otimes \left[\sum_{i_2=1}^{n_2} \sum_{j_2=1}^{n_2} a_{i_2, j_2}^{(2)} |i_2\rangle^{(2)} \langle j_2|^{(2)} \right] \otimes \dots \otimes \left[\sum_{i_N=1}^{n_N} \sum_{j_N=1}^{n_N} a_{i_N, j_N}^{(N)} |i_N\rangle^{(N)} \langle j_N|^{(N)} \right] \\ &= \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \dots \sum_{i_N=1}^{n_N} \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \dots \sum_{j_N=1}^{n_N} \left[a_{i_1, j_1}^{(1)} a_{i_2, j_2}^{(2)} \dots a_{i_N, j_N}^{(N)} \right] |i_1, i_2, \dots, i_N\rangle \langle j_1, j_2, \dots, j_N|. \end{aligned} \quad (2.43)$$

In Mathematica, such an operator tensor product can be calculated similarly to [In\[156\]](#) above. For example, assume that `a1` is a matrix containing the expansion of $\hat{a}^{(1)}$ in its basis, and similarly for `a2` and `a3`. The matrix `A` of expansion coefficients of the full operator $\hat{A} = \hat{a}^{(1)} \otimes \hat{a}^{(2)} \otimes \hat{a}^{(3)}$ is calculated with

```
1 In[157]:=A = KroneckerProduct[a1, a2, a3]
```

Often we need to construct operators which act only on one of the component spaces, as in Eq. (2.36). For example, the operator which generalizes the component Hamiltonians to the full tensor-product Hilbert space is

```
1 In[158]:=H1 = KroneckerProduct[h1,
2           IdentityMatrix[Dimensions[h2]],
3           IdentityMatrix[Dimensions[h3]]];
4 In[159]:=H2 = KroneckerProduct[IdentityMatrix[Dimensions[h1]],
5           h2,
6           IdentityMatrix[Dimensions[h3]]];
7 In[160]:=H3 = KroneckerProduct[IdentityMatrix[Dimensions[h1]],
8           IdentityMatrix[Dimensions[h2]],
9           h3];
10 In[161]:=H = H1 + H2 + H3;
```

where `IdentityMatrix[Dimensions[h1]]` generates a unit matrix of size equal to that of `h1`. In this way, the matrices `H1`, `H2`, `H3` are of equal size and can be added together, even if `h1`, `h2`, `h3` all have different sizes (expressed in Hilbert spaces of different dimensions).

2.4.3 exercises

Q2.6 Two particles of mass m are moving in a three-dimensional harmonic potential $V(r) = \frac{1}{2} m \omega^2 r^2$ with $r = \sqrt{x^2 + y^2 + z^2}$, and interacting via s -wave scattering $V_{\text{int}} = g \delta^3(\vec{r}_1 - \vec{r}_2)$.

1. Write down the Hamiltonian of this system.
2. Propose a basis set in which we can describe the quantum mechanics of this system.
3. Calculate the matrix elements of the Hamiltonian in this basis set.

Q2.7 Calculate `psi` in [In\[156\]](#) (page 39) without using `KroneckerProduct`, but using the `Table` command instead.

Q2.8 Calculate `A` in [In\[157\]](#) (page 40) without using `KroneckerProduct`, but using the `Table` command instead.

Q2.9 Given two spin-1/2 particles in states

$$\begin{aligned} |\psi\rangle^{(1)} &= 0.8|\uparrow\rangle - 0.6|\downarrow\rangle \\ |\psi\rangle^{(2)} &= 0.6i|\uparrow\rangle + 0.8|\downarrow\rangle, \end{aligned} \quad (2.44)$$

use the `KroneckerProduct` function to calculate the joint state $|\psi\rangle = |\psi\rangle^{(1)} \otimes |\psi\rangle^{(2)}$, and compare the result to a manual calculation. In which order do the coefficients appear in the result of `KroneckerProduct`?

Chapter 3

spin systems

In this chapter we put everything we have studied so far together — Mathematica, quantum mechanics, computational bases, units — to study quantum-mechanical systems with finite-dimensional Hilbert spaces. Spin systems are the simplest kind of such systems.

3.1 quantum-mechanical spin and angular momentum operators

As you know, quantum mechanics is not limited to spins (angular momentum) of length $S = 1/2$. A spin (angular momentum) of length S , with $2S \in \mathbb{N}_0$, is represented most easily in the “Dicke basis” of states $|S, M_S\rangle$ with $M_S \in \{S, S-1, S-2, \dots, -S+1, -S\}$. In what follows we will write M instead of M_S whenever no confusion is possible. The operators representing such a spin have the properties

$$\hat{S}_+ |S, M\rangle = \sqrt{S(S+1) - M(M+1)} |S, M+1\rangle \quad (3.1a)$$

$$\hat{S}_- |S, M\rangle = \sqrt{S(S+1) - M(M-1)} |S, M-1\rangle \quad (3.1b)$$

$$\hat{S}_z |S, M\rangle = M |S, M\rangle \quad (3.1c)$$

$$\hat{S}_{\pm} = \hat{S}_x \pm i\hat{S}_y \quad (3.1d)$$

In Mathematica we represent these operators in the Dicke basis as follows, with the elements of the basis set ordered with decreasing projection quantum number M :

```
1 In[162]:=SpinQ[S_] := IntegerQ[2S] && S>=0
2 In[163]:=splus[0] = {{0}} // SparseArray;
3 In[164]:=splus[S_?SpinQ] := splus[S] =
4     SparseArray[Band[{1,2}] -> Table[Sqrt[S(S+1)-M(M+1)],
5     {M,S-1,-S,-1}], {2S+1,2S+1}]
6 In[165]:=sminus[S_?SpinQ] := Transpose[splus[S]]
7 In[166]:=sx[S_?SpinQ] := sx[S] = (splus[S]+sminus[S])/2
8 In[167]:=sy[S_?SpinQ] := sy[S] = (splus[S]-sminus[S])/(2I)
9 In[168]:=sz[S_?SpinQ] := sz[S] =
10     SparseArray[Band[{1,1}] -> Range[S,-S,-1], {2S+1,2S+1}]
11 In[169]:=SparseIdentityMatrix[n_] :=
```

```

12 SparseArray[Band[{1,1}] -> 1, {n,n}]
13 In[170]:=id[S_?SpinQ] := id[S] = SparseIdentityMatrix[2S+1]

```

- Notice that we have defined all these matrix representations as *sparse* matrices (see section 1.7.3), which will make larger calculations much more efficient later on.
- The function `SpinQ[S]` yields `True` only if `S` is a nonnegative half-integer value and can therefore represent a physically valid spin. In general, functions ending in `...Q` are *questions* on the character of an argument: `IntegerQ`, `PrimeQ`, `MemberQ`, `NumericQ`, `EvenQ`, etc. See <http://reference.wolfram.com/mathematica/tutorial/PuttingConstraintsOnPatterns.html> for more information.
- The operator \hat{S}_+ , defined with `splus[S]`, contains only one off-diagonal band of non-zero values. The `SparseArray` matrix constructor allows building such banded matrices by simply specifying the starting point of the band and a vector with the elements of the nonzero band.
- The operator \hat{S}_z , defined with `sz[S]`, shows you the ordering of the basis elements since it has the projection quantum numbers on the diagonal.
- The `IdentityMatrix` function returns a full matrix, which is not suitable for large-scale calculations. It is more efficient to define an equivalent `SparseIdentityMatrix` function which returns a sparse identity matrix of desired size.
- The last operator `id[S]` is the unit operator operating on a spin of length `S`, and will be used below for tensor-product definitions.
- All these matrices can be displayed with, for example,

```

1 In[171]:=sx[3/2] // MatrixForm

```

3.1.1 exercises

- Q3.1** Verify that for $S = 1/2$ the above Mathematica definitions give the Pauli matrices: $\hat{S}_i = \frac{1}{2} \hat{\sigma}_i$ for $i = x, y, z$.
- Q3.2** Verify in Mathematica that $\hat{S}_x^2 + \hat{S}_y^2 + \hat{S}_z^2 = S(S+1) \mathbb{1}$ and $[\hat{S}_x, \hat{S}_y] = i\hat{S}_z$ for several values of S . What is the largest value of S for which you can do this verification within one minute on your computer? *Hint:* use the `Timing` function.
- Q3.3** The operators $\hat{S}_{x,y,z}$ are the generators of rotations: a rotation by an angle α around the axis given by a normalized vector \vec{n} is done with the operator $\hat{R}_{\vec{n}}(\alpha) = \exp(-i\alpha \vec{n} \cdot \vec{\hat{S}})$. Set $\vec{n} = \{\sin(\vartheta) \cos(\varphi), \sin(\vartheta) \sin(\varphi), \cos(\vartheta)\}$ and calculate the operator $\hat{R}_{\vec{n}}(\alpha)$ explicitly for $S = 0$, $S = 1/2$, and $S = 1$. Check that for $\alpha = 0$ you find the unit operator.

3.2 spin-1/2 electron in a dc magnetic field

As a first example we look at a single spin $S = 1/2$. As usual we use the basis containing the two states $|\uparrow\rangle = |\frac{1}{2}, \frac{1}{2}\rangle$ and $|\downarrow\rangle = |\frac{1}{2}, -\frac{1}{2}\rangle$, which we know to be eigenstates of the operators \hat{S}^2 and \hat{S}_z . The matrix expressions of the operators relevant for this system are given by the Pauli matrices divided by two,

$$\mathbf{S}_x = \frac{1}{2} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \frac{1}{2} \boldsymbol{\sigma}_x \quad \mathbf{S}_y = \frac{1}{2} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = \frac{1}{2} \boldsymbol{\sigma}_y \quad \mathbf{S}_z = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \frac{1}{2} \boldsymbol{\sigma}_z \quad (3.2)$$

In Mathematica we enter these as

```
In[172]:=Sx = sx[1/2]; Sy = sy[1/2]; Sz = sz[1/2];
```

using the general definitions of angular momentum operators given in section 3.1.

As a Hamiltonian we use the coupling of this electron spin to an external magnetic field, $\hat{\mathcal{H}} = -\vec{\mu} \cdot \vec{B}$. The magnetic moment of the spin is $\vec{\mu} = -\mu_B g \vec{S}$ in terms of its spin \vec{S} , the Bohr magneton $\mu_B = 9.27400968(20) \times 10^{-24}$ J/T, and the electron's g -factor $g = -2.0023193043622(15)$. The Hamiltonian is therefore

$$\hat{\mathcal{H}} = \mu_B g (\hat{S}_x B_x + \hat{S}_y B_y + \hat{S}_z B_z). \quad (3.3)$$

In our chosen matrix representation this Hamiltonian is

$$\mathbf{H} = \mu_B g (\mathbf{S}_x B_x + \mathbf{S}_y B_y + \mathbf{S}_z B_z) = \frac{1}{2} \mu_B g \begin{pmatrix} B_z & B_x - iB_y \\ B_x + iB_y & -B_z \end{pmatrix}. \quad (3.4)$$

3.2.1 time-independent Schrödinger equation

The time-independent Schrödinger equation for our spin-1/2 problem is, from Eq. (2.14),

$$\frac{1}{2} \mu_B g \begin{pmatrix} B_z & B_x - iB_y \\ B_x + iB_y & -B_z \end{pmatrix} \cdot \vec{\psi} = E \vec{\psi} \quad (3.5)$$

We remember from section 1.9 that most quantities in Eq. 3.5 carry physical units, which the computer cannot deal with. Replacing $B_{x,y,z} = B_0 B'_{x,y,z}$ and $E = E_0 E'$ gives the dimensionless equation

$$\left(\frac{\mu_B B_0}{E_0} \right) \times \frac{g}{2} \begin{pmatrix} B'_z & B'_x - iB'_y \\ B'_x + iB'_y & -B'_z \end{pmatrix} \cdot \vec{\psi} = E' \vec{\psi} \quad (3.6)$$

For concreteness we choose the following units:

magnetic field: $B_0 = 1$ G, a common unit for atomic calculations

energy: $E_0 = h \times 1$ MHz, where $h = 6.62606957 \times 10^{-34}$ Js is Planck's constant. It is common to express energies in units of frequency, where the conversion is sometimes implicitly done via Planck's constant.

We evaluate the numerical prefactor of Eq. (3.6) with

```
1 In[173]:=k = muB*B0/E0 /. {muB -> Quantity["BohrMagneton"],
2           BO -> Quantity["1 Gauss"],
3           EO -> Quantity["PlanckConstant"] *
4           Quantity["1 MHz"]}
5 Out[173]=1.399625
```

The fact that this prefactor k comes out to be of order 1 means that we have chosen an appropriate set of units.

We can now define the Hamiltonian in Mathematica,

```
1 In[174]:=H[Bx_, By_, Bz_] = k * g * (Sx*Bx+Sy*By+Sz*Bz) /.
2 g -> UnitConvert["ElectronGFactor"];
```

and find its eigenvalues (in units of E_0) and eigenvectors:

```
1 In[175]:=Eigensystem[H[Bx,By,Bz]]
```

As described in section 1.7.4 the output is a list with two entries, the first being a list of eigenvalues and the second a list of associated eigenvectors. As long as the Hamiltonian matrix was hermitian, the eigenvalues will all be real-valued; but the eigenvectors can be complex. Since the Hilbert space of this spin problem has dimension 2, and the basis contains two vectors, there are necessarily two eigenvalues and two associated eigenvectors of length 2. The eigenvalues can be called $E_{\pm} = \pm \frac{1}{2} \mu_B g \|\vec{B}\|$, or, in our dimensionless formulation, $E'_{\pm} = \pm k \frac{g}{2} \|\vec{B}'\|$. The list of eigenvalues is given in the Mathematica output as $\{E'_+, E'_-\}$. Notice that these eigenvalues only depend on the magnitude of the magnetic field, and not on its direction. This is to be expected: the choice of the basis as the eigenstates of the \hat{S}_z operator was entirely arbitrary, and therefore the energy eigenvalues cannot depend on the orientation of the magnetic field with respect to this quantization axis. Since there is no preferred axis in this system, there cannot be any directional dependence.

The associated eigenvectors are

$$\vec{\psi}_{\pm} = \left\{ \frac{B_z \pm \|\vec{B}\|}{B_x + iB_y}, 1 \right\}, \quad (3.7)$$

which Mathematica returns as a list of lists, $\{\vec{\psi}_+, \vec{\psi}_-\}$. Notice that these eigenvectors are not normalized.

3.2.2 exercises

Q3.4 Calculate the eigenvalues (in units of J) and eigenvectors (ortho-normalized) of an electron spin in a magnetic field of 1 T in the x -direction.

Q3.5 Set $\vec{B} = B[\vec{e}_x \sin(\vartheta) \cos(\varphi) + \vec{e}_y \sin(\vartheta) \sin(\varphi) + \vec{e}_z \cos(\vartheta)]$ and calculate the eigenvalues and normalized eigenvectors of the electron spin Hamiltonian.

3.3 coupled spin systems: ⁸⁷Rb hyperfine structure

Ground-state Rubidium-87 atoms consist of a nucleus with spin $I = 3/2$, a single valence electron (spin $S = 1/2$, orbital angular momentum $L = 0$, and therefore total spin $J = 1/2$), and 36 core electrons which do not contribute any angular momentum. In a magnetic field along the z -axis, the effective Hamiltonian of this system is¹

$$\hat{\mathcal{H}} = \hat{\mathcal{H}}_0 + hA_{\text{hfs}} \vec{\hat{I}} \cdot \vec{\hat{J}} + \mu_B B_z (g_I \hat{I}_z + g_S \hat{S}_z + g_L \hat{L}_z), \quad (3.8)$$

¹see <http://steck.us/alkalidata/rubidium87numbers.pdf>

where h is Planck's constant, μ_B is the Bohr magneton, $A_{\text{hfs}} = 3.417\,341\,305\,452\,145(45)$ GHz is the spin-spin coupling constant in the ground state of ^{87}Rb , $g_I = -0.000\,995\,141\,4(10)$ is the nuclear g -factor, $g_S = 2.002\,319\,304\,362\,2(15)$ is the electron spin g -factor, and $g_L = 0.999\,993\,69$ is the electron orbital g -factor.

The first part $\hat{\mathcal{H}}_0$ of (3.8) contains all electrostatic interactions, core electrons, nuclear interactions etc. We will assume that the system is in the ground state of $\hat{\mathcal{H}}_0$, which means that the electron is in the $5^2\text{S}_{1/2}$ state. This ground state is eight-fold degenerate and consists of the four magnetic sublevels of the $I = 3/2$ nuclear spin, the two sublevels of the $S = 1/2$ electronic spin, and the single sublevel of the $L = 0$ angular momentum. The basis for the description of this atom is therefore the tensor product basis of a spin-3/2, a spin-1/2, and a spin-0.

The spin operators acting on this composite system are defined as in section 2.4.2. For example, the nuclear-spin operator \hat{I}_x is extended to the composite system by acting trivially on the electron spin and orbital angular momenta, $\hat{I}_x \mapsto \hat{I}_x \otimes \mathbb{1} \otimes \mathbb{1}$. The electron-spin operators are defined accordingly, for example $\hat{S}_x \mapsto \mathbb{1} \otimes \hat{S}_x \otimes \mathbb{1}$. The electron orbital angular momentum operators are, for example, $\hat{L}_x \mapsto \mathbb{1} \otimes \mathbb{1} \otimes \hat{L}_x$. In Mathematica these operators are defined with

```

1 In[176]:=Ix = KroneckerProduct[sx[3/2], id[1/2], id[0]];
2 In[177]:=Iy = KroneckerProduct[sy[3/2], id[1/2], id[0]];
3 In[178]:=Iz = KroneckerProduct[sz[3/2], id[1/2], id[0]];
4 In[179]:=Sx = KroneckerProduct[id[3/2], sx[1/2], id[0]];
5 In[180]:=Sy = KroneckerProduct[id[3/2], sy[1/2], id[0]];
6 In[181]:=Sz = KroneckerProduct[id[3/2], sz[1/2], id[0]];
7 In[182]:=Lx = KroneckerProduct[id[3/2], id[1/2], sx[0]];
8 In[183]:=Ly = KroneckerProduct[id[3/2], id[1/2], sy[0]];
9 In[184]:=Lz = KroneckerProduct[id[3/2], id[1/2], sz[0]];

```

The total electron angular momentum is $\vec{J} = \vec{S} + \vec{L}$:

```

1 In[185]:=Jx = Sx + Lx; Jy = Sy + Ly; Jz = Sz + Lz;

```

The total angular momentum of the ^{87}Rb atom is $\vec{F} = \vec{I} + \vec{J}$:

```

1 In[186]:=Fx = Ix + Jx; Fy = Iy + Jy; Fz = Iz + Jz;

```

From these we can define the hyperfine Hamiltonian with magnetic field in the z -direction as

```

1 In[187]:=Hhf = A(Ix.Jx+Iy.Jy+Iz.Jz) + muB Bz(gI Iz+gS Sz+gL Lz);
2 In[188]:=hfc = {A -> 3417.341305452145,
3               gS -> 2.0023193043622,
4               gL -> 0.99999369,
5               gI -> -0.0009951414,
6               muB -> 1.3996255481168427};

```

where we have made the following assumptions:

- Energies are expressed in units of MHz, after dividing by Planck's constant; magnetic field strengths are expressed in units of Gauss. This is an alternative description of what we did with the constant k in section 3.2.1: essentially we choose a *compatible* system of units which gives $k = 1$ (just like the SI units).

- $A = A_{\text{hfs}}/\text{MHz} = 3417.34$
- $\mu_B = \mu_B/h \times \text{G}/\text{MHz} = 1.399625$:

```

1 In[189]:=UnitConvert["BohrMagneton/PlanckConstant", "MHz/G"]
2 Out[189]=1.399625 MHz/G

```

This yields the Hamiltonian as an 8×8 matrix, and we can calculate its eigenvalues and eigenvectors with

```

1 In[190]:={eval, evec} = Eigensystem[Hhf] // FullSimplify;

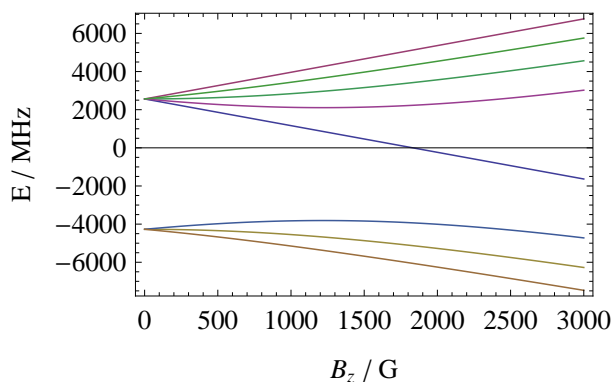
```

We plot the energy eigenvalues with

```

1 In[191]:=Plot[Evaluate[eval /. hfc], {Bz, 0, 3000},
2           Frame -> True, FrameLabel -> {"Bz / G", "E / MHz"}]

```



3.3.1 eigenstate analysis

In this section we analyze the results `eval` and `evec` from the Hamiltonian diagonalization above. For this we first need to define *ortho-normalized* eigenvectors since in general we cannot assume `evec` to be ortho-normalized.

In general we can always define an ortho-normalized eigenvector set with

```

1 In[192]:=nevec = Orthogonalize[evec]

```

The problem with this definition is, however, immediately apparent if you look at the output given by Mathematica: since no assumptions on the reality of the variables were made, the orthogonalization is done in too much generality and quickly becomes unwieldy. Even using `Assuming` and `ComplexExpand`, as in section 1.8, does not give satisfactory results. But if we notice that the eigenvectors in `evec` are all purely real-values, and are already orthogonal, then a simple vector-by-vector normalization is sufficient for calculating an ortho-normalized eigenvector set:

```

1 In[193]:=nevec = #/Sqrt[#. #] & /@ evec;
2 In[194]:=nevec . Transpose[nevec] // FullSimplify

```

The fact that `In[194]` finds a unit matrix implies that the vectors in `nevec` are ortho-normal.

field-free limit

In the field-free limit $B_z = 0$ the energy levels are

```
1 In[195]:= Assuming[A > 0, Limit[eval, Bz -> 0]]
2 Out[195]= {3A/4, 3A/4, -5A/4, 3A/4, -5A/4, 3A/4, -5A/4, 3A/4}
```

We see that the level with energy $-\frac{5}{4}A$ is three-fold degenerate while the level with energy $\frac{3}{4}A$ is five-fold degenerate. This is also visible in the eigenvalue plot above. Considering that we have coupled two spins of lengths $I = \frac{3}{2}$ and $J = \frac{1}{2}$, we expect the composite system to have either total spin $F = 1$ (three sublevels) or $F = 2$ (five sublevels); we can make the tentative assignment that the $F = 1$ level is at energy $E_1 = -\frac{5}{4}A$ and the $F = 2$ level at $E_2 = \frac{3}{4}A$.

In order to demonstrate this assignment we express the matrix elements of the operators \hat{F}^2 and \hat{F}_z in the field-free eigenstates, making sure to normalize these eigenstates before taking the limit $B_z \rightarrow 0$:

```
1 In[196]:= nevec0 = Assuming[A > 0, Limit[nevec, Bz -> 0]];
2 In[197]:= nevec0 . (Fx.Fx+Fy.Fy+Fz.Fz) . Transpose[nevec0]
3 In[198]:= nevec0 . Fz . Transpose[nevec0]
```

Notice that in this calculations we have used the fact that all eigenvectors are real, which may not always be the case for other Hamiltonians. We see that the field-free normalized eigenvectors `nevec0` are eigenvectors of both \hat{F}^2 and \hat{F}_z , and from looking at the eigenvalues we can identify them as

$$\{|2, -2\rangle, |2, 2\rangle, |1, 0\rangle, |2, 0\rangle, |1, -1\rangle, |2, -1\rangle, |1, 1\rangle, |2, 1\rangle\} \quad (3.9)$$

in the notation $|F, M_F\rangle$. These labels are often used to identify the energy eigenstates even for $B_z \neq 0$.

low-field limit

For small magnetic fields, we series-expand the energy eigenvalues to first order in B_z :

```
1 In[199]:= Assuming[A > 0, Series[eval, {Bz, 0, 1}] // FullSimplify]
```

From these low-field terms, in combination with the field-free level assignment, we see that the $F = 1$ and $F = 2$ levels have effective g -factors of $g_1 = -(g_S - 5g_I)/4 \approx -0.501824$ and $g_2 = (g_S + 3g_I)/4 \approx 0.499833$, respectively, so that their energy eigenvalues follow the form

$$E_{F,M_F}(B_z) = E_F(0) + \mu_B M_F g_F B_z + \mathcal{O}(B_z^2). \quad (3.10)$$

These energy shifts due to the magnetic field are called *Zeeman shifts*.

high-field limit

The energy eigenvalues in the high-field limit are infinite; but we can calculate their lowest-order series expansions with

```

1 In[200]:=Assuming[muB > 0 && 0 < -gI < gS,
2       Series[eval, {Bz, Infinity, 1}] // FullSimplify]

```

From these expansions we can already identify the states in the eigenvalue plot above (disregard the terms in $1/B_z$ in the expansions).

In order to calculate the eigenstates in the high-field limit we must again make sure to normalize the states *before* taking the limit $B_z \rightarrow \infty$:

```

1 In[201]:=nevecinf = Assuming[A > 0 && muB > 0 && 0 < -gI < gS,
2       Limit[nevec, Bz -> Infinity]]
3 Out[201]={ {0, 0, 0, 0, 0, 0, 1},
4       {1, 0, 0, 0, 0, 0, 0},
5       {0, 0, 0, -1, 0, 0, 0},
6       {0, 0, 0, 0, 1, 0, 0},
7       {0, 0, 0, 0, 0, -1, 0},
8       {0, 0, 0, 0, 0, 0, 1},
9       {0, -1, 0, 0, 0, 0, 0},
10      {0, 0, 1, 0, 0, 0, 0}}

```

From this we immediately identify the high-field eigenstates as our basis functions in a different order,

$$\{|-\frac{3}{2}, -\frac{1}{2}\rangle, |\frac{3}{2}, \frac{1}{2}\rangle, |\frac{1}{2}, -\frac{1}{2}\rangle, |-\frac{1}{2}, \frac{1}{2}\rangle, |-\frac{1}{2}, -\frac{1}{2}\rangle, |-\frac{3}{2}, \frac{1}{2}\rangle, |\frac{3}{2}, -\frac{1}{2}\rangle, |\frac{1}{2}, \frac{1}{2}\rangle\} \quad (3.11)$$

where we have used the abbreviation $|M_I, M_I\rangle = |\frac{3}{2}, M_I\rangle \otimes |\frac{1}{2}, M_I\rangle$. You can verify this assignment by looking at the matrix elements of the \hat{I}_z and \hat{J}_z operators with

```

1 In[202]:=nevecinf . Iz . Transpose[nevecinf]
2 In[203]:=nevecinf . Jz . Transpose[nevecinf]

```

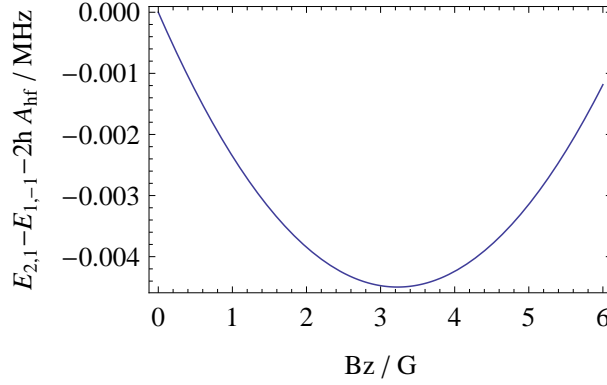
3.3.2 “magic” magnetic field

The energy eigenvalues of the low-field states $|1, -1\rangle$ and $|2, 1\rangle$ have almost the same first-order magnetic field dependence since $g_1 \approx -g_2$ (see low-field limit above). If we plot their energy difference as a function of magnetic field we find an extremal point:

```

1 In[204]:=Plot[eval[[8]]-eval[[5]]-2A /. hfc, {Bz, 0, 6}]

```

At the “magic” field strength $B_0 = 3.22895$ G the energy difference is independent of the magnetic field (to first order):

```

1 In[205]:=FindMinimum[eval[[8]]-eval[[5]]-2A /. hfc, {Bz, 2}]
2 Out[205]={-0.00449737, {Bz -> 3.22895}}

```

3.3.3 coupling to an oscillating magnetic field

In this section we briefly study the coupling of a ⁸⁷Rb atom to a weak oscillating magnetic field. Such a field could be the magnetic part of an electromagnetic wave, whose electric field does not couple to our atom in the electronic ground state. This calculation is a template for more general situations where a quantum-mechanical system is driven by an oscillating field.

The ⁸⁷Rb hyperfine Hamiltonian in the presence of an oscillating magnetic field is

$$\hat{\mathcal{H}}(t) = \underbrace{h A_{\text{hfs}} \vec{\mathbf{I}} \cdot \vec{\mathbf{J}} + \mu_B B_z (g_I \hat{I}_z + g_S \hat{S}_z + g_L \hat{L}_z)}_{\hat{\mathcal{H}}_0} + \cos(\omega t) \times \underbrace{\mu_B \vec{\mathbf{B}}^{\text{ac}} \cdot (g_I \vec{\mathbf{I}} + g_S \vec{\mathbf{S}} + g_L \vec{\mathbf{L}})}_{\hat{\mathcal{H}}_1} \quad (3.12)$$

where the static magnetic field is assumed to be in the z direction, as before. Unfortunately, $[\hat{\mathcal{H}}(t), \hat{\mathcal{H}}(t')] = [\hat{\mathcal{H}}_1, \hat{\mathcal{H}}_0] (\cos(\omega t) - \cos(\omega t')) \neq 0$ in general, so we cannot use the exact solution (2.31) of the time-dependent Schrödinger equation. In fact, the time-dependent Schrödinger equation of this system has no analytic solution at all. In what follows we will calculate approximate solutions.

Since we have diagonalized the time-independent Hamiltonian $\hat{\mathcal{H}}_0$ already, we use its eigenstates as a basis for calculating the effect of the oscillating perturbation $\hat{\mathcal{H}}_1(t)$. In general, calling $\{|i\rangle\}_i$ the set of eigenstates of $\hat{\mathcal{H}}_0$, with $\hat{\mathcal{H}}_0|i\rangle = E_i|i\rangle$, we expand the general hyperfine state as

$$|\psi(t)\rangle = \sum_i \psi_i(t) e^{-iE_i t/\hbar} |i\rangle. \quad (3.13)$$

The time-dependent Schrödinger equation for the expansion coefficients $\psi_i(t)$ in

this interaction picture is given in Eq. (2.28):

$$\begin{aligned} i\hbar\dot{\psi}_i(t) &= \sum_j \psi_j(t) e^{-i(E_j - E_i)t/\hbar} \cos(\omega t) \langle i | \hat{\mathcal{H}}_1 | j \rangle \\ &= \frac{1}{2} \sum_j \psi_j(t) \left[e^{-i\left(\frac{E_j - E_i}{\hbar} - \omega\right)t} + e^{i\left(\frac{E_j - E_i}{\hbar} - \omega\right)t} \right] T_{ij}, \end{aligned} \quad (3.14)$$

where we have replaced $\cos(\omega t) = \frac{1}{2}e^{i\omega t} + \frac{1}{2}e^{-i\omega t}$ and defined

$$T_{ij} = \langle i | \hat{\mathcal{H}}_1 | j \rangle = \langle i | \left[\mu_B \vec{\mathbf{B}}^{\text{ac}} \cdot (g_I \vec{\mathbf{I}} + g_S \vec{\mathbf{S}} + g_L \vec{\mathbf{L}}) \right] | j \rangle. \quad (3.15)$$

From Eq. (3.14) we make two key observations:

Transition matrix elements: The time-independent matrix elements T_{ij} of the perturbation Hamiltonian are called the *transition matrix elements* and describe how the populations of the different eigenstates of $\hat{\mathcal{H}}_0$ are coupled through the oscillating field. We calculate them in Mathematica as follows:

```

1 In[206]:=H0 = A (Ix.Jx + Iy.Jy + Iz.Jz)
2           + muB Bz (gS Sz + gL Lz + gI Iz);
3 In[207]:=H1 = muB (gS (Bacx Sx + Bacx Sy + Bacx Sz)
4           + gI (Bacx Ix + Bacx Iy + Bacx Iz)
5           + gL (Bacx Lx + Bacx Ly + Bacx Lz));
6 In[208]:=H[t_] = H0 + H1 Cos[w t];
7 In[209]:={eval, evec} = Eigensystem[H0] // FullSimplify;
8 In[210]:=nevec = Map[#/Sqrt[#. #] &, evec];
9 In[211]:=T = Assuming[A > 0,
10           nevec.H1.Transpose[nevec] // FullSimplify];

```

Looking at this matrix T we see that not all energy levels are directly coupled by an oscillating magnetic field. For example, $T_{1,2} = 0$ indicates that the populations of the states $|1\rangle$ and $|2\rangle$ can only be indirectly coupled through other states, but not directly.

Numerical solution: We will use the time unit $t_0 = 1 \mu\text{s}$. Since our unit of energy is $E_0 = h \times 1 \text{ MHz}$, the reduced Planck constant takes on the value $\hbar = \hbar / (E_0 t_0) = \hbar / (h \times 1 \text{ MHz} \times 1 \mu\text{s}) = \hbar / h = 1 / (2\pi)$. It is important not to forget this factor in the time-dependent Schrödinger equation.

Eq. (3.14) is a series of linear coupled differential equations, which we can write down explicitly in Mathematica with

```

1 In[212]:=deqs = Table[I*\hbar*Subscript[psi,i]'[t] ==
2           1/2 Sum[Subscript[psi,j][t]*T[[i,j]]*
3           (E^(-I*((eval[[j]]-eval[[i]])/\hbar-w)t) +
4           E^(I*((eval[[i]]-eval[[j]])/\hbar-w)t)),
5           {j,8}], {i,8}] /. \hbar -> 1/(2*Pi);

```

where $w = \omega t_0$ is the frequency of the magnetic field in units of μs^{-1} . Assuming concrete conditions, for example the initial state $|\psi(t=0)\rangle = |F=2, M_F=-2\rangle$

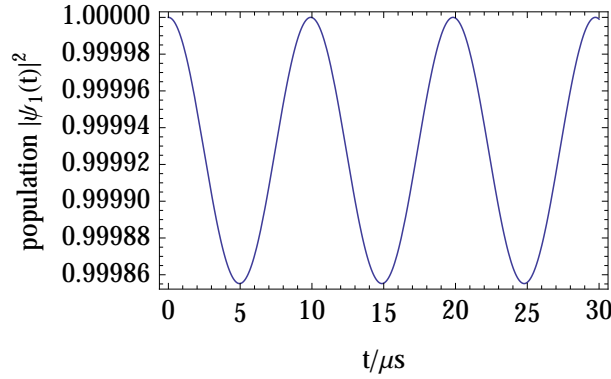
which is the first eigenstate `nevec[[1]]` [see Eq. (3.9)], and magnetic fields $B_z = 3.22895\text{ G}$, $B_x^{\text{ac}} = 1\text{ mG}$, $B_y^{\text{ac}} = B_z^{\text{ac}} = 0$, and an ac field frequency of $\omega = 2\pi \times 6.828\text{ GHz}$, we can find the time-dependent state $|\psi(t)\rangle$ with

```
1 In[213]:=S = NDSolve[Join[deqs/.hfc/.{Bz->3.22895,Bacx->0.001,
2           Bacx->0,Bacz->0,w->2*Pi*6828},
3           {Subscript[psi,1][0]==1,Subscript[psi,2][0]==0,
4           Subscript[psi,3][0]==0,Subscript[psi,4][0]==0,
5           Subscript[psi,5][0]==0,Subscript[psi,6][0]==0,
6           Subscript[psi,7][0]==0,Subscript[psi,8][0]==0},
7           Table[Subscript[psi,i][t],{i,8}], {t, 0, 30},
8           MaxStepSize->10-5, MaxSteps->107]
```

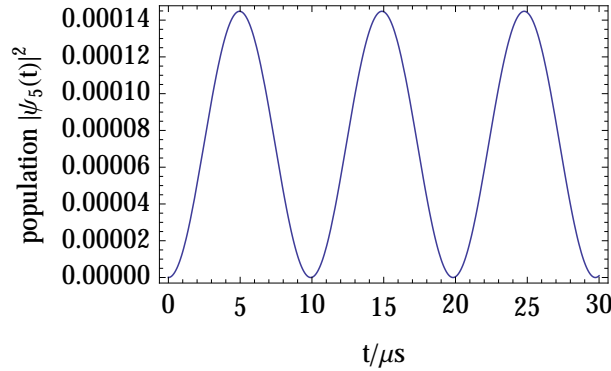
Notice that the maximum step size in this numerical solution is very small ($10^{-5}t_0 = 10\text{ ps}$), since it needs to capture the fast oscillations of more than 6.8 GHz . As a result, a large number of numerical steps is required, which makes this way of studying the evolution very difficult in practice.

We can plot the resulting populations with

```
1 In[214]:=Plot[Abs[Evaluate[Subscript[psi,1][t] /. S[[1]]]]^2,
2           {t, 0, 30}]
```



```
1 In[215]:=Plot[Abs[Evaluate[Subscript[psi,5][t] /. S[[1]]]]^2,
2           {t, 0, 30}]
```



We see that the population is slowly and slightly sloshing between \mathcal{H}_0 -eigenstates $|1\rangle = |F=2, M_F=-2\rangle$ and $|5\rangle = |F=1, M_F=-1\rangle$ [see Eq. (3.9)].

Rotating-wave approximation: The time-dependent prefactor $\exp\left[-i\left(\frac{E_j-E_i}{\hbar}-\omega\right)t\right] + \exp\left[i\left(\frac{E_i-E_j}{\hbar}-\omega\right)t\right]$ of Eq. (3.14) oscillates very rapidly unless either $\frac{E_j-E_i}{\hbar}-\omega \approx 0$ or $\frac{E_i-E_j}{\hbar}-\omega \approx 0$, where one of its terms changes slowly in time. The *rotating-wave approximation* (RWA) consists of neglecting all rapidly rotating terms in Eq. (3.14). Assume that there is a single² pair of states $|i\rangle$ and $|j\rangle$ such that $E_i - E_j \approx \hbar\omega$, while all other states have an energy difference far from $\hbar\omega$. The RWA thus consists of simplifying Eq. (3.14) to

$$\begin{aligned} i\hbar\dot{\psi}_i(t) &\approx \frac{1}{2}\psi_j(t)e^{i\left(\frac{E_i-E_j}{\hbar}-\omega\right)t}T_{ij} \\ i\hbar\dot{\psi}_j(t) &\approx \frac{1}{2}\psi_i(t)e^{-i\left(\frac{E_i-E_j}{\hbar}-\omega\right)t}T_{ji} \\ i\hbar\dot{\psi}_k(t) &\approx 0 \text{ for } k \notin \{i, j\} \end{aligned} \quad (3.16)$$

This approximate system of differential equations has the exact solution

$$\begin{aligned} \psi_i(t) &= e^{-\frac{i}{2}\Delta t} \left[\psi_i(0) \cos \frac{\delta t}{2} + i \left(\frac{\Delta}{\delta} \psi_i(0) - \frac{T_{ij}}{\hbar\delta} \psi_j(0) \right) \sin \frac{\delta t}{2} \right] \\ \psi_j(t) &= e^{\frac{i}{2}\Delta t} \left[\psi_j(0) \cos \frac{\delta t}{2} - i \left(\frac{\Delta}{\delta} \psi_j(0) + \frac{T_{ji}}{\hbar\delta} \psi_i(0) \right) \sin \frac{\delta t}{2} \right] \\ \psi_k(t) &= \psi_k(0) \text{ for } k \notin \{i, j\} \end{aligned} \quad (3.17)$$

in terms of the *detuning* $\Delta = \omega - (E_i - E_j)/\hbar$ and the *generalized Rabi frequency* $\delta = \sqrt{|T_{ij}|^2/\hbar^2 + \Delta^2}$.

On resonance ($\Delta = 0$) these solutions simplify to

$$\begin{aligned} \psi_i(t) &= \psi_i(0) \cos \frac{|T_{ij}|t}{2\hbar} - i \frac{T_{ij}}{|T_{ij}|} \psi_j(0) \sin \frac{|T_{ij}|t}{2\hbar} \\ \psi_j(t) &= \psi_j(0) \cos \frac{|T_{ij}|t}{2\hbar} - i \frac{T_{ij}^*}{|T_{ij}|} \psi_i(0) \sin \frac{|T_{ij}|t}{2\hbar} \\ \psi_k(t) &= \psi_k(0) \text{ for } k \notin \{i, j\} \end{aligned} \quad (3.18)$$

and describe an oscillation of the population between levels $|i\rangle$ and $|j\rangle$ at an angular frequency $\Omega = |T_{ij}|/\hbar$.

Far off-resonance ($\hbar|\Delta| \gg |T_{ij}|$) we have $\delta \approx |\Delta| + |T_{ij}|^2/(2\hbar^2|\Delta|)$, and the solutions (3.17) can be series-expanded to lowest order in $|T_{ij}|/(\hbar|\Delta|)$ as

$$\begin{aligned} \psi_i(t) &\approx e^{i\eta t} \psi_i(0) \\ \psi_j(t) &\approx e^{-i\eta t} \psi_j(0) \\ \psi_k(t) &= \psi_k(0) \text{ for } k \notin \{i, j\} \end{aligned} \quad (3.19)$$

²The following derivation is readily extended to situations where several pairs of states have an energy difference approximately equal to $\hbar\omega$. In such a case we need to solve a larger system of coupled differential equations.

with $\eta = \frac{|T_{ij}|^2}{4\hbar^2\Delta}$. Remember that a rotating phase $e^{-i\epsilon t/\hbar}$ is equivalent to an energy ϵ : this means that the energy of level $|i\rangle$ is effectively shifted to $E'_i = E_i - \hbar\eta$, and that of level $|j\rangle$ is shifted to $E'_j = E_j + \hbar\eta$. For a “blue-detuned” field ($\Delta > 0$) the upper level $|i\rangle$ has a *decreased* energy, whereas the lower level $|j\rangle$ has an *increased* energy. For a “red-detuned” field ($\Delta < 0$) the shifts are reversed. These shifts are called *ac Zeeman shifts* in this case, or *level shifts* more generally.

Assuming that *all* transitions are far off-resonant, the total energy shift of level $|i\rangle$ due to all other levels is

$$E'_i = E_i + \sum_{j \neq i} \text{sign}(E_j - E_i) \frac{|T_{ij}|^2}{4(\hbar\omega - |E_j - E_i|)} \quad (3.20)$$

where $\text{sign}(x) = +1$ for $x > 0$ and $\text{sign}(x) = -1$ for $x < 0$. We calculate these level shifts with the following Mathematica code:

```

1 In[216]:=levelshift = Table[Sum[If[j == i, 0,
2     Sign[eval[[j]]-eval[[i]]] Abs[T[[i,j]]]^2/
3     (4(f-Abs[eval[[j]]-eval[[i]]))],
4     {j, Length[eval]}], {i, Length[eval]}];

```

Here $f = \omega/(2\pi \text{MHz})$ is again the frequency of the magnetic field (in MHz), which is the quantity $\hbar\omega$ in the chosen units of our calculation.

3.3.4 exercises

- Q3.6** Take two angular momenta, for example $I = 3$ and $J = 5$, and calculate the eigenvalues of the operators \hat{I}^2 , \hat{I}_z , \hat{J}^2 , \hat{J}_z , \hat{F}^2 , and \hat{F}_z , where $\vec{F} = \vec{I} + \vec{J}$.
- Q3.7** In Q3.6 you have coupled two angular momenta but you have not used any Clebsch–Gordan coefficients. Why not? Where do these coefficients appear?
- Q3.8** For a spin of a certain length, for example $S = 100$, take the state $|S, S\rangle$ and calculate the expectation values $\langle \hat{S}_x \rangle$, $\langle \hat{S}_y \rangle$, $\langle \hat{S}_z \rangle$, $\langle \hat{S}_x^2 \rangle - \langle \hat{S}_x \rangle^2$, $\langle \hat{S}_y^2 \rangle - \langle \hat{S}_y \rangle^2$, $\langle \hat{S}_z^2 \rangle - \langle \hat{S}_z \rangle^2$.
- Q3.9** Show that the results of the numerical solution plotted with In[214] and In[215] (page 51) can be reproduced with the RWA solution (3.17) with $i = 1$ and $j = 5$.
- Q3.10** Plot the ⁸⁷Rb level shifts at $B_z = 3.22895 \text{ G}$ (the magic field) for the following directions of the oscillating magnetic field:

- circularly polarized around the quantization axis: $\vec{B}^{\text{ac}} = B(\vec{e}_x + i\vec{e}_y)$
- linearly polarized parallel to the quantization axis: $\vec{B}^{\text{ac}} = B\vec{e}_z$

Which polarizations can be absorbed by ⁸⁷Rb at which frequencies?

- Q3.11** Do the presented alkali atom calculation for ²³Na: are there any magic field values?

<http://steck.us/alkalidata/sodiumnumbers.pdf>

Q3.12 Do the presented alkali atom calculation for ^{85}Rb : are there any magic field values?
<http://steck.us/alkalidata/rubidium85numbers.pdf>

Q3.13 Do the presented alkali atom calculation for ^{133}Cs : are there any magic field values?
<http://steck.us/alkalidata/cesiumnumbers.pdf>

3.4 coupled spin systems: transverse Ising model

We now turn to larger numbers of coupled quantum-mechanical spins. A large class of such coupled spin systems can be described with the Hamiltonian

$$\hat{\mathcal{H}} = \sum_{k=1}^N \hat{\mathcal{H}}^{(k)} + \sum_{k=1}^{N-1} \sum_{k'=k+1}^N \hat{\mathcal{H}}_{\text{int}}^{(k,k')}, \quad (3.21)$$

where the $\hat{\mathcal{H}}^{(k)}$ are single-spin Hamiltonians (for example couplings to a magnetic field) and the $\hat{\mathcal{H}}_{\text{int}}^{(k,k')}$ are coupling Hamiltonians between two spins. Direct couplings between three or more spins can usually be neglected.

In particular we study the “transverse Ising” Hamiltonian

$$\hat{\mathcal{H}} = -\frac{b}{2} \sum_{k=1}^N \hat{S}_x^{(k)} - \sum_{k=1}^N \hat{S}_z^{(k)} \hat{S}_z^{(k+1)} \quad (3.22)$$

acting on a ring of N spin- S systems where the $(N+1)$ st spin is identified with the first spin. We can read off three limits from this Hamiltonian:

- For $b \rightarrow \pm\infty$ the spin-spin coupling Hamiltonian can be neglected, and the ground state will have all spins aligned with the $\pm x$ direction,

$$|\psi_{+\infty}\rangle = |\uparrow_x\rangle^{\otimes N}, \quad |\psi_{-\infty}\rangle = |\downarrow_x\rangle^{\otimes N}. \quad (3.23)$$

The system is therefore in a product state for $b \rightarrow \infty$, which means that there is no entanglement between spins. In the basis of $|S, M\rangle$ Dicke states, Eqs. (3.1), the single-spin states making up these product states are

$$|\uparrow_x\rangle = 2^{-S} \sum_{M=-S}^S \sqrt{\binom{2S}{M+S}} |S, M\rangle, \quad |\downarrow_x\rangle = 2^{-S} \sum_{M=-S}^S (-1)^{M+S} \sqrt{\binom{2S}{M+S}} |S, M\rangle, \quad (3.24)$$

which are aligned with the x -axis in the sense that $\hat{S}_x |\uparrow_x\rangle = S |\uparrow_x\rangle$ and $\hat{S}_x |\downarrow_x\rangle = -S |\downarrow_x\rangle$.

- For $b = 0$ the Hamiltonian contains only nearest-neighbor ferromagnetic spin-spin couplings $-\hat{S}_z^{(k)} \hat{S}_z^{(k+1)}$. We know that this Hamiltonian has two degenerate ground states: all spins pointing up or all spins pointing down,

$$|\psi_{0\uparrow}\rangle = |\uparrow_z\rangle^{\otimes N}, \quad |\psi_{0\downarrow}\rangle = |\downarrow_z\rangle^{\otimes N}, \quad (3.25)$$

where in the Dicke-state representation of Eqs. (3.1) we have $|\uparrow_z\rangle = |S, +S\rangle$ and $|\downarrow_z\rangle = |S, -S\rangle$. While these two states are product states, for $|b| \ll 1$ the perturbing Hamiltonian $-\frac{b}{2} \sum_{k=1}^N \hat{S}_x^{(k)}$ is diagonal in the states $\frac{|\psi_{0\uparrow}\rangle \pm |\psi_{0\downarrow}\rangle}{\sqrt{2}}$, which

are *not* product states. The exact ground state for $0 < b \ll 1$ is close to $\frac{|\psi_{01}\rangle + |\psi_{01}\rangle}{\sqrt{2}}$, and for $-1 \ll b < 0$ it is close to $\frac{|\psi_{01}\rangle - |\psi_{01}\rangle}{\sqrt{2}}$. These are both maximally entangled states (“Schrödinger cat states”).

Here we calculate the ground-state wavefunction $|\psi_b\rangle$ as a function of the parameter b , and compare the results to the above asymptotic limits.

3.4.1 basis set

The natural basis set for describing a set of N coupled spins is the tensor-product basis (see section 2.4.2). In this basis, the spin operators $\hat{S}_{x,y,z}^{(k)}$ acting only on spin k are defined as having a trivial action on all other spins, for example

$$\hat{S}_x^{(k)} \mapsto \underbrace{\mathbb{1} \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{(k-1)} \otimes \hat{S}_x \otimes \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{(N-k)}. \quad (3.26)$$

In Mathematica such single-spin- S operators acting on spin k out of a set of N spins are defined with

```

1 In[217]:=op[S_?SpinQ, n_Integer, k_Integer, a_?MatrixQ] /;
2           1<=k<=n && Dimensions[a] == {2S+1,2S+1} :=
3           KroneckerProduct[SparseIdentityMatrix[(2S+1)^(k-1)],
4                             a,
5                             SparseIdentityMatrix[(2S+1)^(n-k)]]
6 In[218]:=sx[S_?SpinQ, n_Integer, k_Integer] /; 1<=k<=n :=
7           op[S, n, k, sx[S]]
8 In[219]:=sy[S_?SpinQ, n_Integer, k_Integer] /; 1<=k<=n :=
9           op[S, n, k, sy[S]]
10 In[220]:=sz[S_?SpinQ, n_Integer, k_Integer] /; 1<=k<=n :=
11           op[S, n, k, sz[S]]

```

Notice that we have used $n = N$ because the symbol N is already used internally in Mathematica.

From these we assemble the Hamiltonian:

```

1 In[221]:=H[S_?SpinQ, n_Integer;/;n>=3, b_] :=
2           -b/2 Sum[sx[S, n, k], {k, n}] -
3           Sum[sz[S, n, k].sz[S, n, Mod[k+1,n,1]], {k, n}]

```

3.4.2 asymptotic ground states

The asymptotic ground states for $b = 0$ and $b \rightarrow \pm\infty$ mentioned above are all product states of the form $|\psi\rangle = |\theta\rangle^{\otimes N}$ where $|\theta\rangle$ is the state of a single spin. We form an N -particle tensor product state of such single-spin states with

```

1 In[222]:=productstate[state_?VectorQ, n_Integer;/;n>=1] :=
2           Flatten[KroneckerProduct @@ Table[state, {n}]]

```

in accordance with [In\[156\]](#) on page 39.

The particular single-spin states $|\uparrow_x\rangle, |\downarrow_x\rangle, |\uparrow_z\rangle, |\downarrow_z\rangle$ we will be using are

```

1 In[223]:=xup[S_?SpinQ] :=
2           2^(-S) Table[Sqrt[Binomial[2S,M+S]],{M,-S,S}]
3 In[224]:=xdn[S_?SpinQ] :=
4           2^(-S) Table[(-1)^(M+S) Sqrt[Binomial[2S,M+S]],{M,-S,S}]
5 In[225]:=zup[S_?SpinQ] := SparseArray[1 -> 1, 2S+1]
6 In[226]:=zdn[S_?SpinQ] := SparseArray[-1 -> 1, 2S+1]

```

3.4.3 Hamiltonian diagonalization

We find the m lowest-energy eigenstates of this Hamiltonian with the procedures described in section 1.7.4: for example, with $S = 1/2$ and $N = 20$,

```

1 In[227]:=With[{S = 1/2, n = 20},
2           (* Hamiltonian *)
3           h[b_] = H[S, n, b];
4           (* two degenerate ground states for b=0 *)
5           gs0up = productstate[zup[S], n];
6           gs0dn = productstate[zdn[S], n];
7           (* ground state for b=+Infinity *)
8           gsplusinf = productstate[xup[S], n];
9           (* ground state for b=-Infinity *)
10          gsminusinf = productstate[xdn[S], n];
11          (* numerically calculate lowest m states *)
12          Clear[gs];
13          gs[b_?NumericQ, m_Integer /; m>=1] := gs[b, m] =
14            -Eigensystem[-h[N[b]], m,
15              Method -> {"Arnoldi", "Criteria" -> "RealPart",
16                MaxIterations -> 10^6}]]

```

Comments:

- $gs0up = |\psi_{0\uparrow}\rangle$ and $gs0dn = |\psi_{0\downarrow}\rangle$ are the exact degenerate ground state wavefunctions for $b = 0$; $gsplusinf = |\psi_{+\infty}\rangle$ and $gsminusinf = |\psi_{-\infty}\rangle$ are the exact nondegenerate ground state wavefunctions for $b = \pm\infty$.
- The function `gs`, which calculates the `m` lowest-lying eigenstates of the Hamiltonian, remembers its calculated values (see section 1.6.3): this is important here because such eigenstate calculations can take a long time.
- The function `gs` numerically calculates the eigenvalues using `h[N[b]]` as a Hamiltonian, which ensures that the Hamiltonian contains floating-point machine-precision numbers instead of exact numbers in case `b` is given as an exact number. Calculating the eigenvalues and eigenvectors of a matrix of exact numbers takes extremely long (please try!).
- When the ground state is degenerate, which happens here for $b = 0$, the Arnoldi algorithm has some difficulty finding the correct degeneracy. This means that `gs[0,2]` may return two non-degenerate eigenstates instead of the (correct)

two degenerate ground states (please try for $N = 15$ and $N = 20$). This is a well-known problem that can be circumvented by calculating more eigenstates: try `gs[0, 10]` and check that the two lowest energy eigenvalues are the same.

- A problem involving N spin- S systems leads to matrices of size $(2S + 1)^N \times (2S + 1)^N$. This scaling quickly becomes very problematic and is at the center of why quantum mechanics is difficult. Imagine a system composed of $N = 1000$ spins $S = 1/2$: its state vector is a list of $2^{1000} = 1.07 \times 10^{301}$ complex numbers! Comparing this to the fact that there are only about 10^{80} particles in the universe, we conclude that such a state vector (wavefunction) could never be written down and therefore the Hilbert space method of quantum mechanics we are using here is fundamentally flawed. But as this is an introductory course, we will stick to this classical matrix-mechanics formalism and let the computer bear the weight of its complexity. Keep in mind, though, that this is not a viable strategy for large systems, as each doubling of computer capacity only allows us to add a single spin to the system, which, using Moore's law, allows us to add one spin every two years.³

There are alternative formulations of quantum mechanics, notably the path-integral formalism, which partly circumvent this problem; but the computational difficulty is not eliminated, it is merely shifted. Modern developments such as matrix-product states try to limit the accessible Hilbert space by limiting calculations to a subspace where the entanglement between particles is bounded. This makes sense since almost all states of the huge Hilbert space are so complex and carry such complicated quantum-mechanical entanglement that (i) they would be extremely difficult to generate with realistic Hamiltonians, and (ii) they would decohere within very short time.

3.4.4 analysis of the ground state

energy gap

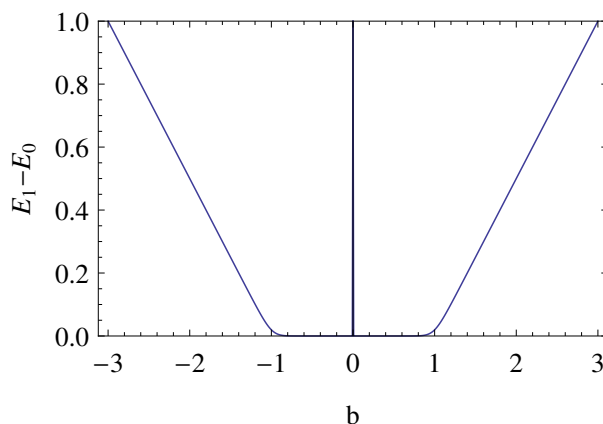
Much of the behavior of our Ising spin chain can be seen in a plot of the *energy gap*, which is the energy difference between the ground state and the first excited state. With `m = 2` we calculate the two lowest-lying energy levels and plot their energy difference as a function of the parameter b :

```

1 In[228]:=With[{bmax = 3, db = 1/128, m = 2},
2           ListLinePlot[Table[{b, gs[b,m][[1,2]]-gs[b,m][[1,1]]},
3           {b, -bmax, bmax, db}]]]

```

³Moore's law is the observation that over the history of computing hardware, the number of transistors on integrated circuits doubles approximately every two years. From http://en.wikipedia.org/wiki/Moore's_law



Even in this small 20-spin simulation we can see that this gap is approximately

$$E_1 - E_0 \approx \begin{cases} 0 & \text{if } |b| < 1, \\ \frac{|b|-1}{2} & \text{if } |b| > 1. \end{cases} \quad (3.27)$$

This observation of a qualitative change in the excitation gap suggests that at $b = \pm 1$ the system undergoes a *quantum phase transition* (i.e., a phase transition induced by quantum fluctuations instead of thermal fluctuations). We note that the gap (3.27) is independent of the particle number N and is therefore a *global* property of the Ising spin ring, not a property of each individual spin (in which case it would scale with N).

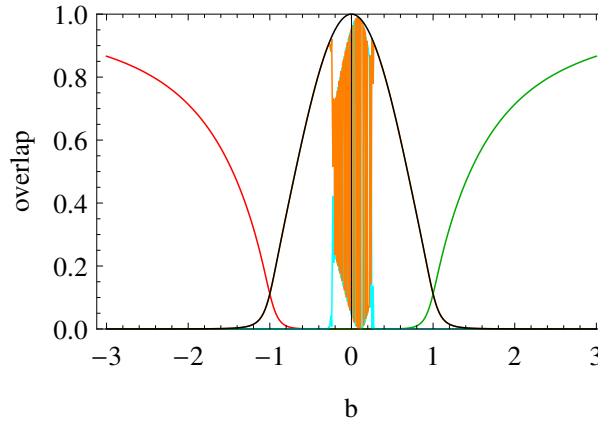
overlap with asymptotic wavefunctions

Once a ground state wavefunction $|\psi_b\rangle$ has been calculated, we compute its overlap with the asymptotically known wavefunctions with scalar products. Notice that for $b = 0$ we calculate the scalar products with the wavefunctions $\frac{|\psi_{01}\rangle \pm |\psi_{0l}\rangle}{\sqrt{2}}$ as they are the approximate ground states for $|b| \ll 1$.

```

1 In[229]:=With[{bmax = 3, db = 1/128, m = 2},
2   ListLinePlot[
3     Table[{b, Abs[gsminusinf.gs[b,m][[2,1]]]^2},
4           {b, Abs[gsplusinf.gs[b,m][[2,1]]]^2},
5           {b, Abs[(gs0up-gs0dn)/Sqrt[2]].gs[b,m][[2,1]]^2},
6           {b, Abs[(gs0up+gs0dn)/Sqrt[2]].gs[b,m][[2,1]]^2},
7           {b, Abs[(gs0up-gs0dn)/Sqrt[2]].gs[b,m][[2,1]]^2 +
8             Abs[(gs0up+gs0dn)/Sqrt[2]].gs[b,m][[2,1]]^2},
9           {b, -bmax, bmax, db}] // Transpose]]

```



Observations:

- The overlap $|\langle \psi_b | \psi_{-\infty} \rangle|^2$ (red) approaches 1 as $b \rightarrow -\infty$.
- The overlap $|\langle \psi_b | \psi_{+\infty} \rangle|^2$ (green) approaches 1 as $b \rightarrow +\infty$.
- The overlap $\left| \langle \psi_b | \frac{|\psi_{0\uparrow}\rangle - |\psi_{0\downarrow}\rangle}{\sqrt{2}} \right|^2$ (cyan) is mostly negligible.
- The overlap $\left| \langle \psi_b | \frac{|\psi_{0\uparrow}\rangle + |\psi_{0\downarrow}\rangle}{\sqrt{2}} \right|^2$ (orange) approaches 1 as $b \rightarrow 0$.
- The sum of these last two, $\left| \langle \psi_b | \frac{|\psi_{0\uparrow}\rangle - |\psi_{0\downarrow}\rangle}{\sqrt{2}} \right|^2 + \left| \langle \psi_b | \frac{|\psi_{0\uparrow}\rangle + |\psi_{0\downarrow}\rangle}{\sqrt{2}} \right|^2 = |\langle \psi_b | \psi_{0\uparrow} \rangle|^2 + |\langle \psi_b | \psi_{0\downarrow} \rangle|^2$ (black), approaches 1 as $b \rightarrow 0$ and is less prone to numerical noise.
- If you redo this calculation with an *odd* number of spins, you may find different overlaps with the $\frac{|\psi_{0\uparrow}\rangle \pm |\psi_{0\downarrow}\rangle}{\sqrt{2}}$ asymptotic wavefunctions. Their sum, however, drawn in black, should be insensitive to the parity of N .
- For $|b| \lesssim 0.2$ the excitation gap (see above) is so small that the calculated ground-state eigenvector is no longer truly the ground state but becomes mixed with the first excited state due to numerical inaccuracies. This leads to the jumps in the orange and cyan curves (notice, however, that their sum, shown in black, is stable). If you redo this calculation with larger values for m , you may get better results.

magnetization

Studying the ground state directly is of limited use because of the large amount of information contained in its numerical representation. We gain more insight by studying specific observables, for example the magnetization $\langle \hat{S}_x^{(k)} \rangle$. We add the following definition to the `With[]` clause in `In[227]` (page 56):

```

1 (* spin components expectation values *)
2 Clear[mx,my,mz];
3 mx[b_?NumericQ, m_Integer /; m >= 1, k_Integer] :=
4   mx[b, m, k] = With[{g = gs[b,m][[2,1]]},

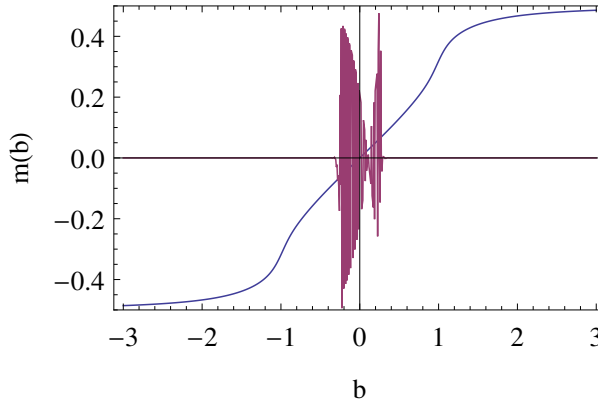
```

```

5      Re[g.(sx[S, n, Mod[k, n, 1]].g)];
6      my[b_?NumericQ, m_Integer /; m >= 1, k_Integer] :=
7      my[b, m, k] = With[{g = gs[b,m][[2,1]]},
8      Re[g.(sy[S, n, Mod[k, n, 1]].g)];
9      mz[b_?NumericQ, m_Integer /; m >= 1, k_Integer] :=
10     mz[b, m, k] = With[{g = gs[b,m][[2,1]]},
11     Re[g.(sz[S, n, Mod[k, n, 1]].g)];

```

In our transverse Ising model only the x -component of the magnetization is nonzero. Due to the translational symmetry of the system we can look at the magnetization of any spin, for example the first one ($k = 1$): $m_x(b)$ (blue) and $m_z(b)$ (red, non-zero due to numerical inaccuracies)



We see that in the phases of large $|b|$, the spins are almost entirely polarized, while in the phase $|b| < 1$ the x -magnetization is roughly proportional to b .

correlations

Another very useful observable is the spin-spin correlation function

$$C_{k,k'} = \langle \vec{\mathcal{S}}^{(k)} \cdot \vec{\mathcal{S}}^{(k')} \rangle - \langle \vec{\mathcal{S}}^{(k)} \rangle \cdot \langle \vec{\mathcal{S}}^{(k')} \rangle. \quad (3.28)$$

For $S = 1/2$ this correlation function has the following known values:

- $-\frac{3}{4} \leq C_{k,k'} \leq +\frac{1}{4}$
- $C_{k,k'} = -\frac{3}{4}$ if the two spins form a singlet, i.e., if they are in the joint state $\frac{|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle}{\sqrt{2}}$. Remember that the spin monogamy theorem states that if spins k and k' form a singlet, then both must be uncorrelated with all other spins in the system.
- $C_{k,k'} = 0$ for uncorrelated spins.
- $C_{k,k'} = +\frac{1}{4}$ for parallel spins, for example in the joint states $\frac{|\uparrow\uparrow\rangle + |\downarrow\downarrow\rangle}{\sqrt{2}}$ or $\frac{|\uparrow\uparrow\rangle - |\downarrow\downarrow\rangle}{\sqrt{2}}$.

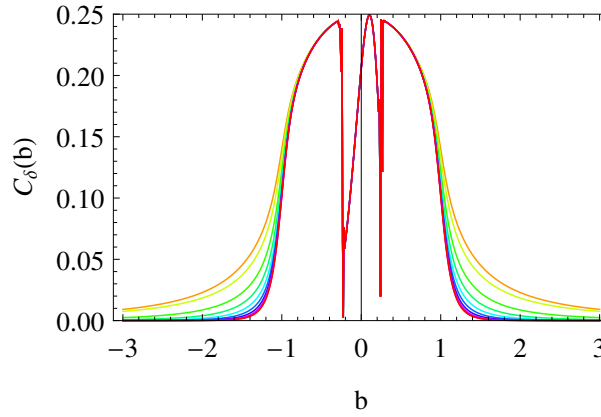
We add the following definition to the `With[]` clause in [In\[227\]](#) (page 56):

```

1  (* spin-spin correlation operator *)
2  Clear[Cop];
3  Cop[k1_Integer, k2_Integer] := Cop[k1, k2] =
4    With[{q1 = Mod[k1,n,1], q2 = Mod[k2,n,1]},
5      sx[S,n,q1].sx[S,n,q2] + sy[S,n,q1].sy[S,n,q2]
6      + sz[S,n,q1].sz[S,n,q2]];
7  (* spin-spin correlations *)
8  Clear[c];
9  c[b_?NumericQ,m_Integer;/;m>=1,{k1_Integer,k2_Integer}] :=
10    c[b,m,{k1,k2}] = With[{g = gs[b,m][[2,1]]},
11      Re[g.(Cop[k1,k2].g)] - (mx[b,m,k1]*mx[b,m,k2]
12      +my[b,m,k1]*my[b,m,k2]+mz[b,m,k1]*mz[b,m,k2])];

```

Since our spin ring is translationally invariant, we can simply plot $C_\delta = C_{1,1+\delta}$: for $N = 20$ and $\delta = 1 \dots 10$ (top to bottom),



Observations:

- The spins are maximally correlated ($C = +\frac{1}{4}$) for $b = 0$, in the ferromagnetic phase. They are all either pointing up or pointing down, so each spin is correlated with each other spin. It is only the spin-spin interactions which correlate the spins' directions and therefore their fluctuations.
- The spins are uncorrelated ($C \rightarrow 0$) for $b \rightarrow \pm\infty$, in the paramagnetic phases. They are all pointing in the $+x$ direction for $b \gg 1$ or in the $-x$ direction for $b \ll -1$, but they are doing so in an independent way and would keep pointing in that direction even if the spin-spin interactions were switched off. This means that the fluctuations of the spins' directions are uncorrelated.

entropy of entanglement

We know now that in the limits $b \rightarrow \pm\infty$ the spins are uncorrelated and polarized, while close to $b = 0$ they are maximally correlated but unpolarized. Here we quantify these correlations with the *entropy of entanglement*, which measures the entanglement of a single spin with the rest of the spin chain.

Assume our quantum-mechanical system can be split into two parts, A and B . In our case, A is the first spin, and B is the rest of the spin ring; but what follows is much more general. Let $\{|i_A\rangle\}$ be a basis set for the description of part A , and $\{|i_B\rangle\}$ a basis set for the description of part B . In all generality the density matrix of the whole system can be expressed as

$$\hat{\rho}_{AB} = \sum_{i_A, i'_A, j_B, j'_B} c_{i_A, j_B, i'_A, j'_B} |i_A\rangle\langle i'_A| \otimes |j_B\rangle\langle j'_B| \quad (3.29)$$

In the case of a pure state $|\psi\rangle$, for example when we calculate a ground state of our Ising ring, the density matrix is $\hat{\rho}_{AB} = |\psi\rangle\langle\psi|$. Since we can represent $|\psi\rangle = \sum_{i_A, j_B} \phi_{i_A, j_B} |i_A\rangle \otimes |j_B\rangle$, the density matrix is

$$\begin{aligned} \hat{\rho}_{AB} = |\psi\rangle\langle\psi| &= \left[\sum_{i_A, j_B} \phi_{i_A, j_B} |i_A\rangle \otimes |j_B\rangle \right] \left[\sum_{i'_A, j'_B} \phi_{i'_A, j'_B}^* \langle i'_A| \otimes \langle j'_B| \right] \\ &= \sum_{i_A, i'_A, j_B, j'_B} \left(\phi_{i_A, j_B} \phi_{i'_A, j'_B}^* \right) |i_A\rangle\langle i'_A| \otimes |j_B\rangle\langle j'_B|, \end{aligned} \quad (3.30)$$

which is of the form of Eq. (3.29).

We define the *reduced density matrix* $\hat{\rho}_A$ of subsystem A by eliminating subsystem B through a *partial trace*:

$$\begin{aligned} \hat{\rho}_A = \text{Tr}_B \hat{\rho}_{AB} &= \sum_{j_B} \langle j_B | \hat{\rho}_{AB} | j_B \rangle = \sum_{i_A, i'_A, j_B, j'_B} c_{i_A, j_B, i'_A, j'_B} |i_A\rangle\langle i'_A| \otimes \langle j_B | j_B \rangle \langle j'_B | j'_B \rangle \\ &= \sum_{i_A, i'_A, j_B} c_{i_A, j_B, i'_A, j_B} |i_A\rangle\langle i'_A|. \end{aligned} \quad (3.31)$$

This density operator only acts on subsystem A and describes its behavior under the assumption that we have no access to observables on subsystem B . For a pure state [Eq. (3.30)], $c_{i_A, j_B, i'_A, j'_B} = \phi_{i_A, j_B} \phi_{i'_A, j'_B}^*$, and the reduced density matrix is

$$\hat{\rho}_A = \sum_{i_A, i'_A, j_B} \phi_{i_A, j_B} \phi_{i'_A, j_B}^* |i_A\rangle\langle i'_A|. \quad (3.32)$$

The entropy of entanglement is defined as the *von Neumann entropy* of the reduced density matrix,

$$S_{AB} = -\text{Tr}(\hat{\rho}_A \log_2 \hat{\rho}_A) = -\sum_i \lambda_i \log_2 \lambda_i \quad (3.33)$$

where the λ_i are the eigenvalues of $\hat{\rho}_A$. Care must be taken with the case $\lambda_i = 0$: we find $\lim_{\lambda \rightarrow 0} \lambda \log_2 \lambda = 0$.

In Mathematica we define the entanglement entropy of the first spin with the rest of the spin ring as follows:

```

1 In[230]:=s[0] = 0; s[x_] = -x Log[2, x];
2 In[231]:=EE[psi_] := Module[{g, rhoA},
3   (* compute the single-spin reduced density matrix *)
4   g = Transpose[Partition[psi, 2]];
5   rhoA = Conjugate[g].Transpose[g];
6   (* compute entropy of entanglement *)
7   Total[s /@ Re[Eigenvalues[rhoA]]]

```

Observations:

- Entanglement entropies of the known asymptotic ground states:

```

1 In[232]:=EE[(gs0up+gs0dn)/Sqrt[2]]
2 Out[232]=1
3 In[233]:=EE[(gs0up-gs0dn)/Sqrt[2]]
4 Out[233]=1
5 In[234]:=EE[gsplusinf]
6 Out[234]=0
7 In[235]:=EE[gsminusinf]
8 Out[235]=0

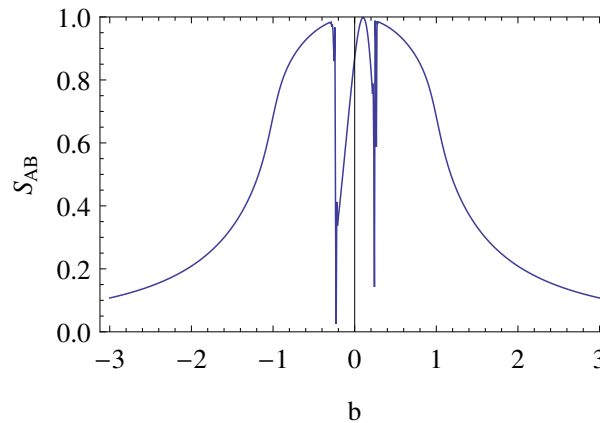
```

- Entanglement entropy as a function of b : again the calculation is numerically difficult around $b \approx 0$ because of the quasi-degeneracy.

```

1 In[236]:=With[{bmax = 3, db = 1/128, m = 2},
2   ListLinePlot[Table[{b, EE[gs[b,m][[2,1]]}],
3     {b, -bmax, bmax, db}], PlotRange -> {0, 1}]

```



Notice that the quantum phase transition is not visible in this plot.

3.4.5 exercises

- Q3.14** Show that the single-spin states of Eq. (3.24), implemented in [In\[223\]](#) and [In\[224\]](#), are indeed eigenstates of \hat{S}_x with eigenvalues $\pm S$.
- Q3.15** For $S = 1/2$, what is the largest value of N for which you can calculate the ground-state wavefunction of the transverse Ising model at the critical point $b = 1$?
- Q3.16** Study the transverse Ising model with $S = 1$:
1. At which values of b do you find quantum phase transitions?
 2. Characterize the ground state in terms of magnetization, spin-spin correlations, and entanglement entropy.

Q3.17 Study the transverse XY model for $S = 1/2$:

$$\hat{\mathcal{H}} = -\frac{b}{2} \sum_{k=1}^N \hat{S}_z^{(k)} - \sum_{k=1}^N \left(\hat{S}_x^{(k)} \hat{S}_x^{(k+1)} + \hat{S}_y^{(k)} \hat{S}_y^{(k+1)} \right) \quad (3.34)$$

1. Guess the shape of the ground-state wavefunctions for $b \pm \infty$ [notice that the first term in the Hamiltonian (3.34) is in the z -direction!] and compare to the numerical calculations.
2. At which values of b do you find quantum phase transitions?
3. Characterize the ground state in terms of magnetization, spin-spin correlations, and entanglement entropy.

Q3.18 Do the same calculation for $S = 1/2$ with the Heisenberg-interaction Hamiltonian

$$\hat{\mathcal{H}} = -\frac{b}{2} \sum_{k=1}^N \hat{S}_z^{(k)} - \sum_{k=1}^N \vec{\mathbf{S}}^{(k)} \cdot \vec{\mathbf{S}}^{(k+1)} \quad (3.35)$$

1. Guess the shape of the ground-state wavefunctions for $b \pm \infty$ [notice that the first term in the Hamiltonian (3.34) is in the z -direction!] and compare to the numerical calculations.
2. What is the ground-state degeneracy for $b = 0$?
3. At which values of b do you find quantum phase transitions?
4. Characterize the ground state in terms of magnetization, spin-spin correlations, and entanglement entropy.

Q3.19 Consider two spin-1/2 particles in the triplet state $|\psi\rangle = |\uparrow\uparrow\rangle$. Subsystem A is the first spin, and subsystem B is the second spin.

1. What is the density matrix $\hat{\rho}_{AB}$ of this system?
2. What is the reduced density matrix $\hat{\rho}_A$ of subsystem A (the first spin)? Is this a pure state? If yes, what state?
3. What is the reduced density matrix $\hat{\rho}_B$ of subsystem B (the second spin)? Is this a pure state? If yes, what state?
4. Calculate the von Neumann entropies of $\hat{\rho}_{AB}$, $\hat{\rho}_A$, and $\hat{\rho}_B$.

Q3.20 Consider two spin-1/2 particles in the singlet state $|\psi\rangle = \frac{|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle}{\sqrt{2}}$. Subsystem A is the first spin, and subsystem B is the second spin.

1. What is the density matrix $\hat{\rho}_{AB}$ of this system?
2. What is the reduced density matrix $\hat{\rho}_A$ of subsystem A (the first spin)? Is this a pure state? If yes, what state?
3. What is the reduced density matrix $\hat{\rho}_B$ of subsystem B (the second spin)? Is this a pure state? If yes, what state?
4. Calculate the von Neumann entropies of $\hat{\rho}_{AB}$, $\hat{\rho}_A$, and $\hat{\rho}_B$.

Chapter 4

real-space systems

4.1 one particle in one dimension

One-dimensional single-particle systems are governed by Hamiltonians of the form

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(x). \quad (4.1)$$

The system's behavior is determined by the mass m and the potential $V(x)$.

In what follows we restrict the freedom of the particle to a domain $x \in \Omega = [0, a]$, where a can be very large in order to approximately describe quasi-infinite systems. This assumes the potential to be

$$V(x) = \begin{cases} \infty & \text{for } x \leq 0 \\ W(x) & \text{for } 0 < x < a \\ \infty & \text{for } x \geq a \end{cases} \quad (4.2)$$

This restriction is necessary in order to achieve a finite representation of the system in a computer.

4.1.1 basis functions

The Hilbert space of this particle consists of all square-integrable (L^2) and differentiable wavefunctions with support in Ω . For each ket $|\psi\rangle$ in this Hilbert space we define the wavefunction $\psi(x) = \langle x|\psi\rangle$ in terms of the “position basis” $\{|x\rangle\}_{x \in \Omega}$, which satisfies the completeness relation $\int_0^a |x\rangle \langle x| dx = \mathbb{1}_\Omega$.¹ The scalar product in Ω is defined as

$$\langle \psi|\chi\rangle = \langle \psi| \left[\int_0^a |x\rangle \langle x| dx \right] |\chi\rangle = \int_0^a \langle \psi|x\rangle \langle x|\chi\rangle dx = \int_0^a \psi^*(x) \chi(x) dx. \quad (4.3)$$

As usual we need a set of basis functions $\{|i\rangle\}_i$ to describe this system. There are many possible choices of basis functions. The position basis $\{|x\rangle\}_{x \in \Omega}$ is ill suited

¹To be exact, the position basis set $\{|x\rangle\}_{x \in \Omega}$ spans a space that is much larger than the Hilbert space of square-integrable smooth functions used in quantum mechanics. This can be seen by noting that this basis set has an uncountable number of elements, while the dimension of the Hilbert space in question is only countably infinite [see Eq. (4.4) for a countably infinite basis set]. For example, the state $\sum_{x \in (\Omega \cap \mathbb{Q})} |x\rangle$ is not a valid quantum-mechanical state (it is too pathological), yet it can be expressed in this position basis.

for this task, since its elements are singular and therefore difficult to represent in a computer. The most generally useful ones are the *momentum basis* and the *finite-resolution position basis*, which we will look at in turn, and which will be shown to be related to each other by a type-I discrete sine transform.

momentum basis

The simplest one-dimensional quantum-mechanical system of the type of Eq. (4.1) is the infinite square well with $W(x) = 0$. Its energy eigenstates are

$$\langle x|n\rangle = \phi_n(x) = \sqrt{\frac{2}{a}} \sin\left(\frac{n\pi x}{a}\right) \quad (4.4)$$

for $n = 1, 2, 3, \dots$, with eigen-energies

$$E_n = \frac{n^2 \pi^2 \hbar^2}{2ma^2}. \quad (4.5)$$

We know from the Sturm–Liouville theorem that these functions form a complete set (see Q2.2 on page 33); further, we can use Mathematica to show that they are ortho-normalized:

```

1 In[237]:=phi[a_, n_, x_] = Sqrt[2/a] Sin[n Pi x/a];
2 In[238]:=Table[Integrate[phi[a,n1,x]*phi[a,n2,x], {x, 0, a}],
3           {n1, 0, 10}, {n2, 0, 10}] // MatrixForm

```

They are eigenstates of the squared momentum operator $\hat{p}^2 = \left(-i\hbar \frac{\partial}{\partial x}\right)^2 = -\hbar^2 \frac{\partial^2}{\partial x^2}$:

$$\hat{p}^2|n\rangle = \frac{n^2 \pi^2 \hbar^2}{a^2}|n\rangle. \quad (4.6)$$

This makes the kinetic-energy operator $\hat{\mathcal{H}}_{\text{kin}} = \hat{p}^2/(2m)$ diagonal in this basis: $\langle n|\hat{\mathcal{H}}_{\text{kin}}|n'\rangle = E_n \delta_{nn'}$. However, in general the potential energy, and most other important terms which will appear later, are difficult to express in this momentum basis.

The momentum basis (4.4) contains a countably infinite number of basis functions. In practical calculations, we restrict the computational basis to $n \in \{1 \dots n_{\text{max}}\}$, which means that we only consider physical phenomena with excitation energies below $E_{n_{\text{max}}} = \frac{n_{\text{max}}^2 \pi^2 \hbar^2}{2ma^2}$.

finite-resolution position basis

Given an energy-limited momentum basis set $\{|n\rangle\}_{n=1}^{n_{\text{max}}}$, we define a set of n_{max} equally-spaced points

$$x_j = a \times \frac{j}{n_{\text{max}} + 1} \quad (4.7)$$

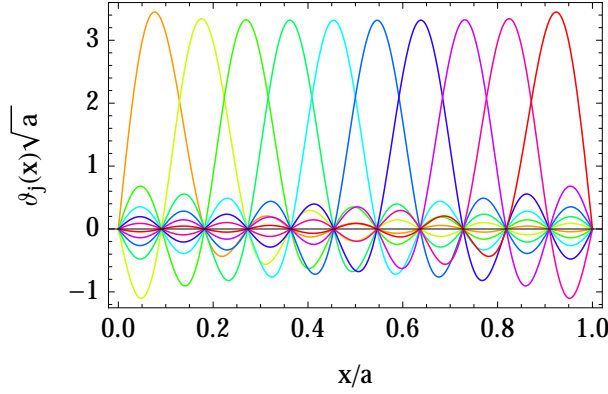
for $j \in \{1 \dots n_{\text{max}}\}$. We then define a new basis set as the closest possible representations of delta-functions at these points:

$$|j\rangle = \sqrt{\frac{a}{n_{\text{max}} + 1}} \sum_{n=1}^{n_{\text{max}}} \phi_n(x_j)|n\rangle. \quad (4.8)$$

The spatial wavefunctions of these basis kets are

$$\langle x|j\rangle = \vartheta_j(x) = \sqrt{\frac{a}{n_{\max} + 1}} \sum_{n=1}^{n_{\max}} \phi_n(x_j) \phi_n(x). \quad (4.9)$$

Here is an example of what these position-basis functions look like for $n_{\max} = 10$:



This new basis set is also ortho-normal, $\langle j|j'\rangle = \delta_{jj'}$, and it is strongly local in the sense that only the basis function $\vartheta_j(x)$ is nonzero at x_j , while all others vanish:

$$\langle x_{j'}|j\rangle = \vartheta_j(x_{j'}) = \delta_{jj'} \times \sqrt{\frac{n_{\max} + 1}{a}}. \quad (4.10)$$

We define these basis functions in Mathematica with

```

1 In[239]:=nmax = 10;
2 In[240]:=xx[a_, j_] = a j/(nmax+1);
3 In[241]:=theta[a_, j_, x_] =
4     Sqrt[a/(nmax+1)] Sum[phi[a,n,xx[a,j]] phi[a,n,x],
5     {n, 1, nmax}];

```

Since the basis function $\vartheta_j(x)$ is the only one which is nonzero at x_j , and it is close to zero everywhere else (exactly zero at the $x_{j' \neq j}$), we can usually make two approximations:

- If a wavefunction is given in the position basis, $|\psi\rangle = \sum_{j=1}^{n_{\max}} v_j |j\rangle$, then by

Eq. (4.10) the wavefunction is known at the grid points, $\psi(x_j) = v_j \times \sqrt{\frac{n_{\max} + 1}{a}}$. This allows for very easy plotting of wavefunctions and densities by linearly interpolating between these grid points:

```

1 In[242]:=ListLinePlot[
2     Transpose[{Table[xx[j], {j, 1, nmax}],
3     (nmax+1)/a * Abs[v]^2}]]

```

By the truncation of the basis at n_{\max} , the wavefunction has no frequency components faster than one half-wave per grid-point spacing, and therefore we can be sure that this linear interpolation is a reasonably accurate representation of the full density $|\langle x|\psi\rangle|^2$, in particular as $n_{\max} \rightarrow \infty$.

- If a potential energy function $W(x)$ varies smoothly over length scales of $x_{j+1} - x_j = a/(n_{\max} + 1)$, then the matrix elements of this potential energy in the position basis are approximately diagonal,

$$\begin{aligned} V_{jj'} &= \langle j | \hat{V} | j' \rangle = \langle j | \left[\int_0^a |x\rangle \langle x| dx \right] \hat{V} \left[\int_0^a |x'\rangle \langle x'| dx' \right] | j' \rangle \\ &= \int_0^a dx \int_0^a dx' \langle j | x \rangle \langle x | \hat{V} | x' \rangle \langle x' | j' \rangle = \int_0^a dx \vartheta_j^*(x) W(x) \vartheta_{j'}(x) \\ &\approx W(x_j) \int_0^a dx \vartheta_j^*(x) \vartheta_{j'}(x) = \delta_{jj'} W(x_j), \end{aligned} \quad (4.11)$$

where we have used Eq. (4.2) and the fact that $\langle x | \hat{V} | x' \rangle = W(x) \delta(x - x')$ since the potential is diagonal in the position basis, as well as the approximate locality of $\vartheta_j(x)$ around x_j implying $W(x) \vartheta_j(x) \approx W(x_j) \vartheta_j(x)$. This is a massive simplification compared to the explicit evaluation of potential integrals for each specific potential energy function.

conversion between basis sets

Within the approximation of a truncation at maximum energy $E_{n_{\max}}$, we can express any wavefunction $|\psi\rangle$ in both basis sets (4.4) and (4.9):

$$|\psi\rangle = \sum_{n=1}^{n_{\max}} u_n |n\rangle = \sum_{j=1}^{n_{\max}} v_j |j\rangle \quad (4.12)$$

Inserting the definition of Eq. (4.8) into Eq. (4.12) we find

$$\begin{aligned} \sum_{n=1}^{n_{\max}} u_n |n\rangle &= \sum_{j=1}^{n_{\max}} v_j \left[\sqrt{\frac{a}{n_{\max} + 1}} \sum_{n'=1}^{n_{\max}} \phi_{n'}(x_j) |n'\rangle \right] \\ &= \sum_{n'=1}^{n_{\max}} \left[\sqrt{\frac{a}{n_{\max} + 1}} \sum_{j=1}^{n_{\max}} v_j \phi_{n'}(x_j) \right] |n'\rangle \end{aligned} \quad (4.13)$$

and therefore, since the basis set $\{|n\rangle\}$ is ortho-normalized,

$$u_n = \sqrt{\frac{a}{n_{\max} + 1}} \sum_{j=1}^{n_{\max}} v_j \phi_n(x_j) = \sum_{j=1}^{n_{\max}} X_{nj} v_j \quad (4.14)$$

with the basis conversion coefficients

$$X_{nj} = \sqrt{\frac{a}{n_{\max} + 1}} \phi_n(x_j) = \sqrt{\frac{a}{n_{\max} + 1}} \sqrt{\frac{2}{a}} \sin\left(\frac{n\pi x_j}{a}\right) = \sqrt{\frac{2}{n_{\max} + 1}} \sin\left(\frac{\pi n j}{n_{\max} + 1}\right). \quad (4.15)$$

The inverse transformation is found from $|n\rangle = \sum_{j=1}^{n_{\max}} \langle j | n \rangle |j\rangle$ inserted into Eq. (4.12), giving

$$v_j = \sum_{n=1}^{n_{\max}} X_{nj} u_n \quad (4.16)$$

in terms of the same coefficients (4.15). Thus the transformations relating the vectors \vec{u} (with components u_n) and \vec{v} (with components v_j) are $\vec{v} = \mathbf{X} \cdot \vec{u}$ and $\vec{u} = \mathbf{X} \cdot \vec{v}$ in terms of the same symmetric matrix \mathbf{X} with coefficients X_{nj} .

We could calculate these coefficients with

```

1 In[243]:=X = Table[Sqrt[2/(nmax+1)] Sin[Pi*n*j/(nmax+1)],
2           {n, 1, nmax}, {j, 1, nmax}] // N;

```

but this is not very efficient, especially for large n_{\max} .

It turns out that Eq. (4.14) and (4.16) relate the vectors \vec{u} and \vec{v} by a *type-I discrete sine transform* (DST-I), which Mathematica can evaluate very efficiently via a fast Fourier transform.² Since the DST-I is its own inverse, we can use

```

1 In[244]:=v = FourierDST[u, 1];
2 In[245]:=u = FourierDST[v, 1];

```

to effect such conversions. We will see a very useful application of this transformation when we study the time-dependent behavior of a particle in a potential (“split-step method”, section 4.1.3).

The matrix X is calculated most efficiently by repeated calls to the DST-I function:

```

1 In[246]:=SparseIdentityMatrix[n_] :=
2           SparseArray[Band[{1,1}]->1, {n,n}]
3 In[247]:=X = FourierDST[#, 1] & /@ SparseIdentityMatrix[nmax];

```

This matrix notation of the basis transformation is useful for converting *operator* representations between the basis sets: the momentum representation U and the position representation V of the same operator satisfy

```

1 In[248]:=V = X.U.X;
2 In[249]:=U = X.V.X;

```

This easy conversion is very useful for the construction of the matrix representations of Hamiltonian operators, since the kinetic energy is diagonal in the momentum basis, Eq. (4.6), while the potential energy operator is approximately diagonal in the position basis, Eq. (4.11).

special case: the kinetic energy operator

The representation of the kinetic energy operator can be calculated exactly with the description given above. Using Eq. (4.6), the kinetic Hamiltonian is

$$\hat{\mathcal{H}}_{\text{kin}} = \frac{\hat{p}^2}{2m} \approx \frac{1}{2m} \left[\sum_{n=1}^{n_{\max}} |n\rangle\langle n| \right] \hat{p}^2 \left[\sum_{n'=1}^{n_{\max}} |n'\rangle\langle n'| \right] = E_1 \sum_{n=1}^{n_{\max}} n^2 |n\rangle\langle n|, \quad (4.17)$$

where $E_1 = \frac{\pi^2 \hbar^2}{2ma^2}$ [see Eq. (4.5)]. In Mathematica, we define the kinetic energy operator in the momentum basis as

```

1 In[250]:=HkinM = h^2 Pi^2/(2 m a^2) *
2           SparseArray[Band[{1,1}]->Table[n^2, {n,1,nmax}]]

```

²see http://en.wikipedia.org/wiki/Fast_Fourier_transform

From this we can calculate the representation in the finite-resolution position basis with

```
In[251]:=HkinP = X . HkinM . X
```

However, for large n_{\max} it is often acceptable to use the following approximation:

$$\langle j | \hat{\mathcal{H}}_{\text{kin}} | j' \rangle \approx E_1 \times \begin{cases} \frac{n_{\max}(n_{\max}+2)}{3} & \text{if } j = j', \\ (-1)^{j-j'} \times \frac{2n_{\max}(n_{\max}+2)}{\pi^2(j-j')^2} & \text{if } j \neq j'. \end{cases} \quad (4.18)$$

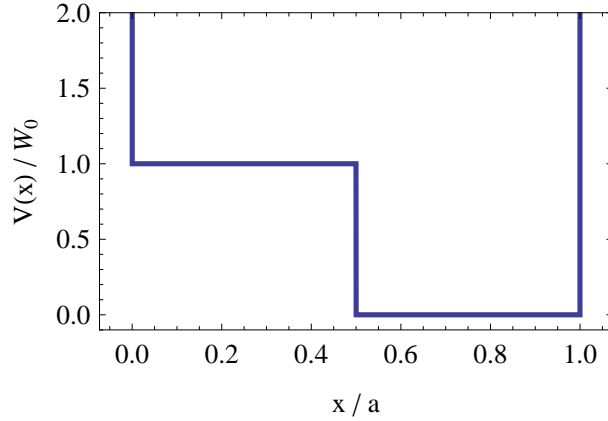
We will not be using this approximation in what follows, as the basis-set conversion through the matrix \mathbf{X} is usually sufficiently efficient.

4.1.2 example: square well with bottom step

A simple example you may remember from quantum-mechanics class is a particle moving in the one-dimensional potential given by Eq. (4.2) with

$$W(x) = \begin{cases} W_0 & \text{if } x < \frac{a}{2} \\ 0 & \text{if } x \geq \frac{a}{2} \end{cases} \quad (4.19)$$

where we assume $W_0 \geq 0$ for simplicity; the case $W_0 \leq 0$ is solved in the exact same fashion.



analytic solution for $0 \leq E \leq W_0$

The potential (4.19) is so simple that we can find the eigenstates of this particle analytically. Since the potential is piecewise flat, we know that the energy eigenstates must be (hyperbolic) sine and cosine functions with piecewise constant wavelengths. In order to find these wavelengths we set

$$\begin{aligned} \psi_1(x) &= A \sinh \left[k_1 \pi \frac{x}{a} \right] & \text{for } 0 < x \leq \frac{a}{2} \\ \psi_2(x) &= B \sin \left[k_2 \pi \left(1 - \frac{x}{a} \right) \right] & \text{for } \frac{a}{2} \leq x < a \end{aligned} \quad (4.20)$$

which satisfy $\psi_1(0) = \psi_2(a) = 0$ to match the boundary conditions where the potential becomes infinite. We assume that $k_1, k_2 \geq 0$.

The conditions for matching these two pieces of the wavefunction are $\psi_1(\frac{a}{2}) = \psi_2(\frac{a}{2})$ and $\psi_1'(\frac{a}{2}) = \psi_2'(\frac{a}{2})$, from which we find the condition

$$k_1 \coth \frac{\pi k_1}{2} = -k_2 \cot \frac{\pi k_2}{2}. \quad (4.21)$$

The time-independent Schrödinger equation further supplies the energy condition

$$E = W_0 - \frac{\hbar^2 \pi^2 k_1^2}{2ma^2} = \frac{\hbar^2 \pi^2 k_2^2}{2ma^2}. \quad (4.22)$$

Since we have assumed that $0 \leq E \leq W_0$ we find from this that $k_1 = \sqrt{\Omega - k_2^2}$ in terms of the dimensionless parameter

$$\Omega = \frac{W_0}{E_1} = \frac{2ma^2 W_0}{\pi^2 \hbar^2}. \quad (4.23)$$

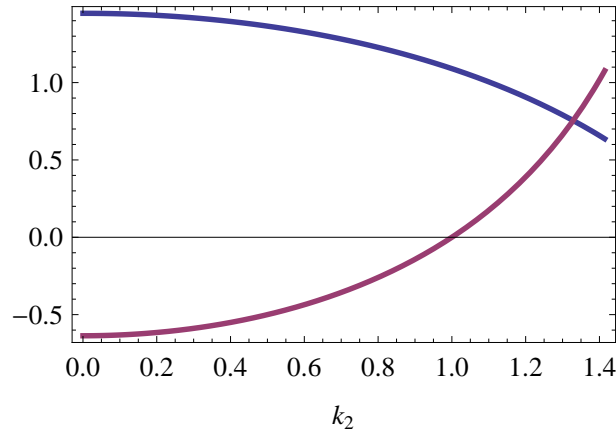
We notice that the entire problem only depends on this one dimensionless parameter Ω , and not on the individual values of m , a , and W_0 : the effort of making the problem dimensionless has paid off by significantly reducing the number of parameters that we need to study. The resulting eigenvalue equation

$$\sqrt{\Omega - k_2^2} \coth \frac{\pi \sqrt{\Omega - k_2^2}}{2} = -k_2 \cot \frac{\pi k_2}{2}. \quad (4.24)$$

thus depends only on one parameter Ω , and can be solved graphically for k_2 in the range $0 \leq k_2 \leq \sqrt{\Omega}$.

For $\Omega < 1.66809$ there is no solution for k_2 , meaning that the ground state has energy $E > W_0$. As a numerical example, for $\Omega = 2$ we plot the left-hand side of Eq. (4.24) in blue and the right-hand side in red:

```
1 In[252]:=With[{Omega = 2},
2   Plot[{Sqrt[Omega-k2^2] Coth[Pi Sqrt[Omega-k2^2]/2],
3     -k2 Cot[Pi k2/2]}, {k2, 0, Sqrt[Omega]}]
```



We find a single solution for the ground state at $k_2 = 1.32884$ numerically with

```

1 In[253]:=s = With[{Omega = 2},
2           FindRoot[Sqrt[Omega-k2^2] Coth[Pi Sqrt[Omega-k2^2]/2]
3                   == -k2 Cot[Pi k2/2], {k2, 1}]]
4 Out[253]={k2 -> 1.32884}

```

Notice that the result is given as a list of replacement rules (with the `->` operator). You can extract the value of k_2 with

```

1 In[254]:=k2 /. s
2 Out[254]=1.32884

```

and calculate the value of k_1 with

```

1 In[255]:=With[{Omega = 2},
2 In[256]:= Sqrt[Omega-k2^2] /. s
3 Out[256]=0.48392

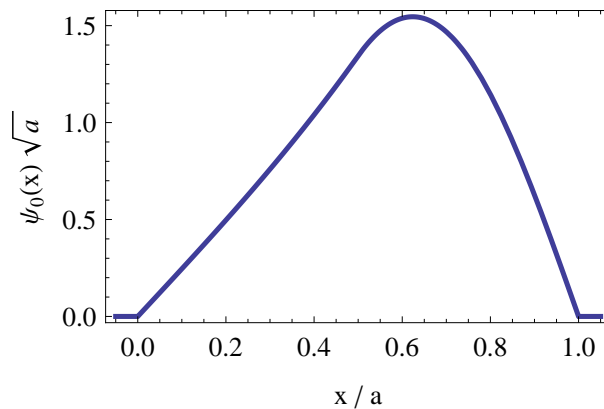
```

We can plot the result with (assuming $a = 1$)

```

1 In[257]:=With[{k1=0.4839202839634602, k2=1.3288420368007343,
2             A=1.6088142613650431, B=1.5458263302568298},
3           psi0[x_] = Piecewise[{{A Sinh[k1 Pi x], 0<=x<=1/2},
4                                {B Sin[k2 Pi (1-x)], 1/2<x<=1}}];
5           Plot[psi0[x], {x, 0, 1}, Exclusions->None]]

```



We will be using this wavefunction `psi0[x]` below for a comparison with numerical calculations.

For $E > W_0$ the same calculation must be re-done with $\psi_1(x) = A \sin(k_1 x/a)$. The algebra is very similar, and the results do not teach us anything further for this course.

exercises

Q4.1 Find and plot the ground state for $\Omega = 1000$. What is the probability to find the particle in the left half of the potential well?

numerical solution (I): momentum basis

We first search for the ground state of the step-well in the momentum basis. The matrix elements of the kinetic energy are diagonal,

$$\langle n | \hat{\mathcal{H}}_{\text{kin}} | n' \rangle = \frac{n^2 \pi^2 \hbar^2}{2ma^2} \times \delta_{nn'}. \quad (4.25)$$

The matrix elements of the potential energy (4.19) are

$$\begin{aligned} \langle n | \hat{V} | n' \rangle &= \int_0^a \phi_n^*(x) W(x) \phi_{n'}(x) dx \\ &= \frac{2W_0}{a} \int_0^{\frac{a}{2}} \sin\left(\frac{n\pi x}{a}\right) \sin\left(\frac{n'\pi x}{a}\right) dx = 2W_0 \int_0^{\frac{1}{2}} \sin(n\pi y) \sin(n'\pi y) dy \\ &= W_0 \times \begin{cases} \frac{1}{2} & \text{if } n = n' \\ \frac{1}{\pi} \left[\frac{(-1)^{\frac{n+n'+1}{2}}}{n+n'} - \frac{(-1)^{\frac{n-n'+1}{2}}}{n-n'} \right] & \text{if } n + n' \text{ odd} \\ 0 & \text{otherwise} \end{cases} \quad (4.26) \end{aligned}$$

This allows us to express the matrix elements of the Hamiltonian $H_{nn'} = \langle n | \hat{\mathcal{H}} | n' \rangle$ in units of the energy $E_1 = \frac{\pi^2 \hbar^2}{2ma^2}$:

$$\frac{H_{nn'}}{E_1} = n^2 \delta_{nn'} + \Omega \times \begin{cases} \frac{1}{2} & \text{if } n = n' \\ \frac{1}{\pi} \left[\frac{(-1)^{\frac{n+n'+1}{2}}}{n+n'} - \frac{(-1)^{\frac{n-n'+1}{2}}}{n-n'} \right] & \text{if } n + n' \text{ odd} \\ 0 & \text{otherwise} \end{cases} \quad (4.27)$$

with $\Omega = W_0/E_1$ the same dimensionless parameter as above. In Mathematica,

```
1 In[258]:=h[Omega_, n_, n_] = n^2 + Omega/2;
2 In[259]:=h[Omega_, n_, np_] /; OddQ[n+np] = Omega/Pi *
3           ((-1)^((n+np+1)/2)/(n+np) - (-1)^((n-np+1)/2)/(n-np));
4 In[260]:=h[Omega_, n_, np_] /; EvenQ[n+np] = 0;
```

For a given n_{max} we can now find the Hamiltonian matrix with

```
1 In[261]:=nmax = 10;
2 In[262]:=H[Omega_] = Table[h[Omega,n,np], {n,1,nmax}, {np,1,nmax}];
```

and the ground state coefficients with

```
1 In[263]:=Clear[gs];
2 In[264]:=gs[Omega_?NumericQ] := gs[Omega] =
3           -Eigensystem[-H[N[Omega]], 1,
4           Method -> {"Arnoldi", "Criteria" -> "RealPart",
5           MaxIterations -> 10^6}]
```

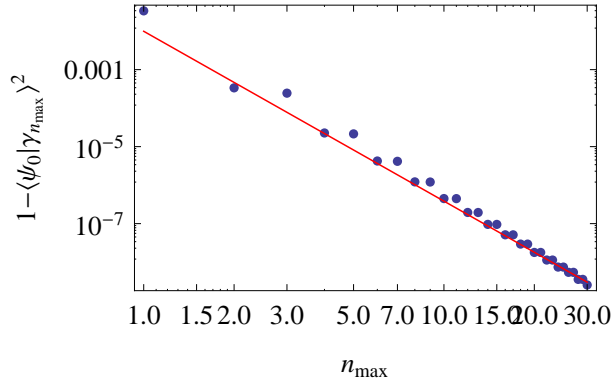
The ground state wavefunction $\langle x | \gamma_{n_{\text{max}}} \rangle$ is then

```
1 In[265]:=psi[Omega_?NumericQ, a_, x_] :=
2           gs[Omega][[2,1]] . Table[phi[a, n, x], {n, 1, nmax}]
```

For $\Omega = 2$ we can calculate the overlap of this numerical ground state with the exact one given in [In\[257\]](#), $\langle \psi_0 | \gamma_{n_{\max}} \rangle$:

```
1 In[266]:=NIntegrate[psi0[x]*psi[2,1,x], {x,0,1}]
```

This overlap quickly approaches unity as n_{\max} increases:



The red line is a least-squares fit to the even values of n_{\max} , giving a convergence of $1 - \langle \psi_0 | \gamma_{n_{\max}} \rangle^2 \approx 0.0099 n_{\max}^{-4.41}$.

numerical solution (II): mixed basis

The main difficulty of the first numerical solution above was the evaluation of the potential matrix elements (4.26). For such a simple step potential as used here, we were able to find an analytic expression for $\langle n | \hat{V} | n' \rangle$; but for more complicated potentials this will not be possible. But we have seen in Eq. (4.11) that the potential is approximately diagonal in the finite-resolution position basis, and we can therefore find an approximate expression for the Hamiltonian matrix with a procedure that is independent of the shape of $W(x)$.

We first calculate the matrix elements of the kinetic energy operator in the momentum basis, again in units of E_1 :

```
1 In[267]:=nmax = 10;
2 In[268]:=HkinM = SparseArray[Band[{1,1}]->Table[n^2, {n,1,nmax}]];
```

Next we convert this operator into the finite-resolution position basis:

```
1 In[269]:=SparseIdentityMatrix[n_] :=
2     SparseArray[Band[{1,1}]->1, {n,n}]
3 In[270]:=X = FourierDST[#, 1] & /@ SparseIdentityMatrix[nmax];
4 In[271]:=HkinP = X . HkinM . X;
```

The potential energy operator \hat{W} is expressed approximately in the finite-resolution position basis: setting $a = 1$ for simplicity and using $\Omega = W_0/E_1$,

```
1 In[272]:=W[Omega_, x_] = Piecewise[{{Omega, x<1/2},
2     {Omega/2, x==1/2}, {0, x>1/2}}];
```

```

3 In[273]:=xval = Table[j/(nmax+1), {j, 1, nmax}];
4 In[274]:=HpotP[Omega_] = SparseArray[Band[{1,1}] ->
5           (W[Omega, #]& /@ xval)];

```

The full Hamiltonian in the finite-resolution position basis is

```

1 In[275]:=HP[Omega_] = HkinP + HpotP[Omega];

```

We find the ground state with

```

1 In[276]:=Clear[gsP];
2 In[277]:=gsP[Omega_?NumericQ] := gsP[Omega] =
3           -Eigensystem[-HP[N[Omega]], 1,
4             Method -> {"Arnoldi", "Criteria" -> "RealPart",
5             MaxIterations -> 10^6}]

```

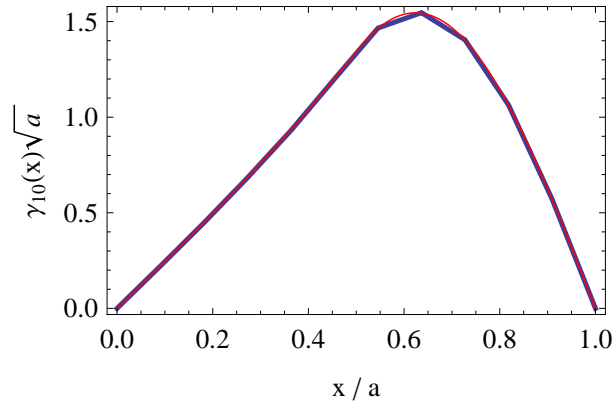
and, as shown before, this can be plotted simply with

```

1 In[278]:=With[{Omega=2},
2           gammaP = Join[
3             {{0,0}},
4             Transpose[{xval, Sqrt[nmax+1]*gsP[Omega][[2,1]]}],
5             {{1,0}}];
6           ListLinePlot[gammaP]

```

where we have “manually” added the known values $\gamma(0) = \gamma(1) = 0$ to the list of numerically calculated wave-function values.



You can see that even with $n_{\max} = 10$ grid points this ground-state wavefunction (thick blue line) looks remarkably close to the exact one (thin red line, see page 72).

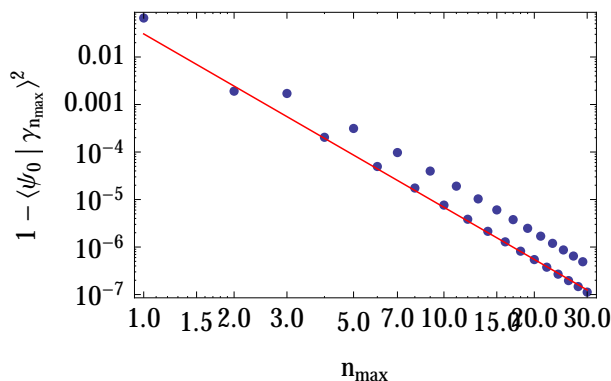
The wavefunction is calculated by converting to the momentum representation as in [In\[245\]](#) and multiplying with the basis functions as in [In\[265\]](#):

```

1 In[279]:=psiP[Omega_?NumericQ, a_, x_] :=
2           FourierDST[gsP[Omega][[2,1]], 1] .
3           Table[phi[a,n,x], {n,1,nmax}]

```

As for [In\[266\]](#) the overlap of this numerical wavefunction with the exact one approaches unity as n_{\max} increases:



The red line is a least-squares fit to the even values of n_{\max} , giving a convergence of $1 - \langle \psi_0 | \gamma_{n_{\max}} \rangle^2 \approx 0.0306178 n_{\max}^{-3.64889}$. This convergence is slower than for the momentum-basis calculation since there was an additional approximation involved in Eq. (4.11).

exercises

Q4.2 Find and plot the ground state for $\Omega = 1000$, using the approximate numerical method. What is the probability to find the particle in the left half of the potential well?

Q4.3 Calculate the energy levels and energy eigenstates of a particle in a well with bottom potential

$$W(x) = \frac{1}{2}k \left(x - \frac{1}{2} \right)^2 \quad (4.28)$$

Compare them to the analytically known eigen-energies and eigenstates of a harmonic oscillator.

Q4.4 With $a = 1$, take a “noisy” potential $W(x) = \Omega \times \sum_{n=1}^{\hat{n}} \alpha_n \phi_n(x)$ with α_n random: $\langle \alpha_n \rangle = 0$ and $\langle \alpha_n^2 \rangle = n^{-2}$. Plot the ground-state density $|\gamma(x)|^2$ using $n_{\max} \gg \hat{n}$, for different values of Ω .

4.1.3 dynamics

Assume again a single particle of mass m moving in a one-dimensional potential, with Hamiltonian

$$\hat{\mathcal{H}} = \underbrace{-\frac{\hbar^2}{2m} \frac{d^2}{dx^2}}_{\hat{\mathcal{H}}_{\text{kin}}} + \underbrace{V(x)}_{\hat{\mathcal{H}}_{\text{pot}}}. \quad (4.29)$$

The motion is again restricted to $x \in [0, a]$. We want to study the time-dependent wavefunction $\psi(x, t) = \langle x | \psi(t) \rangle$ given in Eq. (2.32) on page 37,

$$|\psi(t)\rangle = \exp \left[-\frac{i(t-t_0)}{\hbar} \hat{\mathcal{H}} \right] |\psi(t_0)\rangle. \quad (4.30)$$

The simplest way of computing this propagation is to express the wavefunction and the Hamiltonian in a particular basis and use a matrix exponentiation to find the time dependence of the expansion coefficients of the wavefunction. For example, if we use the finite-resolution position basis, we have seen on page 74 how to find the matrix representation of the Hamiltonian, [HP](#). For a given initial wavefunction `psi0` we can then define

```
In[280]:=psi[t_?NumericQ] := MatrixExp[-I*t*HP].psi0
```

where we have changed the units such that the time $t = (t - t_0)/\hbar$ is in units of inverse energy. If you try this out, you will see that calculating $|\psi(t)\rangle$ in this way is not very efficient, because the matrix exponentiation is a numerically difficult operation.

A much more efficient method can be found by first splitting up the Hamiltonian as $\hat{\mathcal{H}} = \hat{\mathcal{H}}_{\text{kin}} + \hat{\mathcal{H}}_{\text{pot}}$ as in Eq. (4.29), and then using the Trotter expansion

$$e^{\lambda(X+Y)} = e^{\frac{\lambda}{2}X} e^{\lambda Y} e^{\frac{\lambda}{2}X} \times e^{\frac{\lambda^3}{24}[X,[X,Y]] + \frac{\lambda^3}{12}[Y,[X,Y]]} \times e^{-\frac{\lambda^4}{48}[X,[X,[X,Y]]] - \frac{\lambda^4}{16}[X,[Y,[X,Y]]] - \frac{\lambda^4}{24}[Y,[Y,[X,Y]]]} \dots$$

$$\approx e^{\frac{\lambda}{2}X} e^{\lambda Y} e^{\frac{\lambda}{2}X}, \quad (4.31)$$

where the approximation is valid for small λ since the neglected terms are of third and higher orders in λ (notice that there is no second-order term in λ !). Setting $\lambda = -\frac{i(t-t_0)}{M\hbar}$ for some large integer M , as well as $X = \hat{\mathcal{H}}_{\text{pot}}$ and $Y = \hat{\mathcal{H}}_{\text{kin}}$, we find

$$\begin{aligned} |\psi(t)\rangle &= \lim_{M \rightarrow \infty} \left[e^{\lambda \hat{\mathcal{H}}} \right]^M |\psi(t_0)\rangle = \lim_{M \rightarrow \infty} \left[e^{\lambda(\hat{\mathcal{H}}_{\text{kin}} + \hat{\mathcal{H}}_{\text{pot}})} \right]^M |\psi(t_0)\rangle \\ &\stackrel{\text{Trotter}}{=} \lim_{M \rightarrow \infty} \left[e^{\frac{\lambda}{2} \hat{\mathcal{H}}_{\text{pot}}} e^{\lambda \hat{\mathcal{H}}_{\text{kin}}} e^{\frac{\lambda}{2} \hat{\mathcal{H}}_{\text{pot}}} \right]^M |\psi(t_0)\rangle \\ &= \lim_{M \rightarrow \infty} e^{\frac{\lambda}{2} \hat{\mathcal{H}}_{\text{pot}}} \underbrace{e^{\lambda \hat{\mathcal{H}}_{\text{kin}}} e^{\lambda \hat{\mathcal{H}}_{\text{pot}}} e^{\lambda \hat{\mathcal{H}}_{\text{kin}}} e^{\lambda \hat{\mathcal{H}}_{\text{pot}}} \dots e^{\lambda \hat{\mathcal{H}}_{\text{kin}}} e^{\frac{\lambda}{2} \hat{\mathcal{H}}_{\text{pot}}}}_{(M-1) \text{ repetitions of } e^{\lambda \hat{\mathcal{H}}_{\text{kin}}} e^{\lambda \hat{\mathcal{H}}_{\text{pot}}}} |\psi(t_0)\rangle. \quad (4.32) \end{aligned}$$

This can be evaluated very efficiently. We express the potential Hamiltonian in the finite-resolution position basis, the kinetic Hamiltonian in the momentum basis, and the time-dependent wavefunction in both bases (4.12):

$$|\psi(t)\rangle = \sum_{n=1}^{n_{\text{max}}} u_n(t) |n\rangle = \sum_{j=1}^{n_{\text{max}}} v_j(t) |j\rangle \quad (4.33a)$$

$$\hat{\mathcal{H}}_{\text{pot}} = \sum_{j=1}^{n_{\text{max}}} W(x_j) |j\rangle \langle j| \quad (4.33b)$$

$$\hat{\mathcal{H}}_{\text{kin}} = \sum_{n=1}^{n_{\text{max}}} n^2 |n\rangle \langle n| \quad (4.33c)$$

where we have already expressed all energies as multiples of the square-well ground-state energy $E_1 = \frac{\pi^2 \hbar^2}{2ma^2}$. The expansion coefficients of the wavefunction are related by a discrete Fourier transform (4.16).

The matrix exponential of a diagonal matrix is easily found from the Taylor ex-

pansion:

$$\begin{aligned} e^{\lambda \hat{\mathcal{H}}_{\text{pot}}} &= \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} \hat{\mathcal{H}}_{\text{pot}}^k = \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} \left[\sum_{j=1}^{n_{\max}} W(x_j) |j\rangle \langle j| \right]^k = \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} \left[\sum_{j=1}^{n_{\max}} W^k(x_j) |j\rangle \langle j| \right] \\ &= \sum_{j=1}^{n_{\max}} \left[\sum_{k=0}^{\infty} \frac{\lambda^k}{k!} W^k(x_j) \right] |j\rangle \langle j| = \sum_{j=1}^{n_{\max}} e^{\lambda W(x_j)} |j\rangle \langle j|, \quad (4.34) \end{aligned}$$

where we have used the integer matrix powers

$$\begin{aligned} &\left[\sum_{j=1}^{n_{\max}} W(x_j) |j\rangle \langle j| \right]^k \\ &= \left[\sum_{j_1=1}^{n_{\max}} W(x_{j_1}) |j_1\rangle \langle j_1| \right] \left[\sum_{j_2=1}^{n_{\max}} W(x_{j_2}) |j_2\rangle \langle j_2| \right] \cdots \left[\sum_{j_k=1}^{n_{\max}} W(x_{j_k}) |j_k\rangle \langle j_k| \right] \\ &= \sum_{j_1=1}^{n_{\max}} \sum_{j_2=1}^{n_{\max}} \cdots \sum_{j_k=1}^{n_{\max}} W(x_{j_1}) W(x_{j_2}) \cdots W(x_{j_k}) |j_1\rangle \langle j_1| |j_2\rangle \langle j_2| \cdots |j_k\rangle \langle j_k| \\ &= \sum_{j_1=1}^{n_{\max}} \sum_{j_2=1}^{n_{\max}} \cdots \sum_{j_k=1}^{n_{\max}} W(x_{j_1}) W(x_{j_2}) \cdots W(x_{j_k}) \delta_{j_1, j_2} \delta_{j_2, j_3} \cdots \delta_{j_{k-1}, j_k} |j_k\rangle \langle j_k| \\ &= \sum_{j=1}^{n_{\max}} W^k(x_j) |j\rangle \langle j|. \quad (4.35) \end{aligned}$$

The action of the potential Hamiltonian thus becomes straightforward:

$$e^{\lambda \hat{\mathcal{H}}_{\text{pot}}} |\psi(t)\rangle = \left[\sum_{j=1}^{n_{\max}} e^{\lambda W(x_j)} |j\rangle \langle j| \right] \left[\sum_{j'=1}^{n_{\max}} v_{j'}(t) |j'\rangle \right] = \sum_{j=1}^{n_{\max}} \left[e^{\lambda W(x_j)} v_j(t) \right] |j\rangle, \quad (4.36)$$

which is a simple element-by-element multiplication of the coefficients of the wavefunction with the exponentials of the potential – no matrix operations are required. The expansion coefficients (position basis) after propagation with the potential Hamiltonian are therefore

$$v'_j = e^{\lambda W(x_j)} v_j. \quad (4.37)$$

The action of the kinetic Hamiltonian in the momentum representation is found in the exactly same way:

$$e^{\lambda \hat{\mathcal{H}}_{\text{kin}}} |\psi(t)\rangle = \left[\sum_{n=1}^{n_{\max}} e^{\lambda n^2} |n\rangle \langle n| \right] \left[\sum_{n'=1}^{n_{\max}} u_{n'}(t) |n'\rangle \right] = \sum_{n=1}^{n_{\max}} \left[e^{\lambda n^2} u_n(t) \right] |n\rangle. \quad (4.38)$$

The expansion coefficients (momentum basis) after propagation with the kinetic Hamiltonian are therefore

$$u'_n = e^{\lambda n^2} u_n. \quad (4.39)$$

We know that a type-I discrete sine transform brings the wavefunction from the finite-resolution position basis to the momentum basis (4.16). The propagation under the kinetic Hamiltonian thus consists of

1. a type-I discrete sine transform to calculate the coefficients $v_j \mapsto u_n$,
2. an element-by-element multiplication (4.39) to find the coefficients $u_n \mapsto u'_n$,

3. and a second type-I discrete sine transform to calculate the coefficients $u'_n \mapsto v'_j$.

Here we assemble all these pieces into a program which propagates a state $|\psi(t_0)\rangle$ given as a coefficient vector \vec{v} in the finite-resolution basis forward in time to $t = t_0 + \Delta t$ with M time steps. We assume $a = 1$ and $E_1 = 1$.

```

1 In[281]:=HpotP = SparseArray[Band[{1,1}]->Wval];
2 In[282]:=HkinM = SparseArray[Band[{1,1}]->Table[n^2,{n,1,nmax}]];
3 In[283]:=HkinP = X . HkinM . X;
4 In[284]:=HP = HkinP + HpotP;
5 In[285]:=propExact[Dt_?NumericQ, psi_?(VectorQ[#,NumericQ]&)] :=
6     MatrixExp[-I*Dt*HP].psi
7 In[286]:=propApprox[Dt_?NumericQ, M_Integer/;M>=2,
8     psi_?(VectorQ[#,NumericQ]&)] :=
9     Module[{Ke,Pe2,Pe,psi1,psi2,propKin,propPot},
10        (* compute exponentials *)
11        Ke = Exp[-I*Dt/M*Table[n^2,{n,1,nmax}]];
12        Pe2 = Exp[-I/2*Dt/M*Wval];
13        Pe = Pe2^2;
14        (* propagate with the potential operator *)
15        propPot[p_] := Pe * p;
16        (* propagate with the kinetic operator *)
17        propKin[p_] := FourierDST[Ke*FourierDST[p,1],1];
18        (* propagation *)
19        psi1 = propKin[Pe2*psi];
20        psi2 = Nest[propKin[propPot[#]]&, psi1, M-1];
21        Pe2*psi2]
```

Notice that there are no basis functions, integrals, etc. involved in this calculation; everything is done in terms of the values of the wavefunction on the grid $x_1 \dots x_{n_{\max}}$. This efficient method is called *split-step propagation*.

The `Nest` command “nests” a function call: for example, `Nest[f,x,3]` calculates $f(f(f(x)))$. We use this on line 20 above to repeatedly propagate by the potential and kinetic operators. This propagation algorithm can be adapted to calculate the wavefunction at all the intermediate times $t = t_0 + \frac{m}{M}(t - t_0)$ for $m = 1, 2, 3, \dots, M$, which allows us to follow the evolution of the wavefunction during its time evolution. To achieve this we simply replace the `Nest` command with `NestList`, which is similar to `Nest` but returns all intermediate results: for example, `NestList[f,x,3]` calculates the list $\{x, f(x), f(f(x)), f(f(f(x)))\}$. We replace the code above from line 20 with

```

20     psi2 = NestList[propKin[propPot[#]] &, psi1, M-1];
21     Transpose[{Range[0, M]*Dt/M,
22         Prepend[(Pe2*#) & /@ psi2, psi]}]]
```

exercises

Q4.5 Convince yourself that the Trotter expansion (4.31) is really necessary, i.e., that $e^{X+Y} \neq e^X e^Y$ if X and Y do not commute. *Hint:* use two concrete non-

commuting objects X and Y , for example two random 2×2 matrices as generated with `RandomReal[{0, 1}, {2, 2}]`.

Q4.6 Given a particle moving in the range $x \in [0, 1]$ with the scaled Hamiltonian

$$\hat{\mathcal{H}} = -\frac{1}{\pi^2} \frac{d^2}{dx^2} + \Omega \sin(10\pi x), \quad (4.40)$$

compute its time-dependent wavefunction starting from $\psi(t=0) \propto e^{-\frac{(x-1/2)^2}{4\sigma^2}} e^{ikx}$ with $\sigma = 0.05$ and $k = 100$. Compute $\langle x \rangle(t)$ for $t = 0 \dots 0.2$ using first $\Omega = 0$ and then $\Omega = 1000$.

4.2 Many particles in one dimension: dynamics with the non-linear Schrödinger equation

The advantage of the split-step evolution (4.32) becomes particularly clear if the particle evolves according to the *non-linear* Hamiltonian

$$\hat{\mathcal{H}} = \underbrace{-\frac{\hbar^2}{2m} \frac{d^2}{dx^2}}_{\hat{\mathcal{H}}_{\text{kin}}} + \underbrace{V(x) + g|\psi(x)|^2}_{\hat{\mathcal{H}}_{\text{pot}}}. \quad (4.41)$$

Such Hamiltonians can describe the *mean-field interactions* between N particles which are all in wavefunction $\psi(x)$, and which are therefore in a joint product wavefunction $\psi(x)^{\otimes N}$. One particle's wavefunction $\psi(x)$ (normalized to $\int |\psi(x, t)|^2 dx = 1$) sees a potential generated by the average density $(N-1)|\psi(x)|^2$ of other particles with the same wavefunction, usually through collisional interactions. The associated non-linear Schrödinger equation is called the *Gross-Pitaevskii equation* and describes the dynamics of Bose-Einstein condensates:

$$i\hbar \frac{\partial \psi(x, t)}{\partial t} = \left[-\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(x) + g|\psi(x, t)|^2 \right] \psi(x, t). \quad (4.42)$$

The coefficient $g = (N-1) \times \frac{4\pi\hbar^2 a_s}{m}$ approximates the mean-field s -wave scattering between a particle and the $(N-1)$ other particles, with s -wave scattering length a_s .

For any $g \neq 0$ there is no solution of the form (4.30). But the split-step method of Eq. (4.32) can still be used because the potential is still diagonal in the position representation. We extend the Mathematica code of the previous section by modifying the `propApprox` method to include a non-linear term with prefactor `g`, and do not forget that the wavefunction at grid point x_j is $\psi(x_j) = \sqrt{\frac{n_{\text{max}}+1}{a}} \times v_j$:

```

1 In[287]:=propApprox[Dt_?NumericQ, M_Integer/;M>=2, g_?NumericQ,
2           psi_?(VectorQ[#, NumericQ]&)] :=
3           Module[{Ke, psi1, psi2, propKin, propPot},
4             (* compute exponentials *)
5             Ke = Exp[-I*Dt/M*Table[n^2, {n, 1, nmax}]];
6             (* propagate with the potential operator *)
7             propPot[dt_, p_] :=
8             Exp[-I*dt*(Wval+g*(nmax+1)*Abs[p]^2)] * p;

```



```

9      (* propagate with the kinetic operator *)
10     propKin[p_] := FourierDST[Ke*FourierDST[p,1],1];
11     (* propagation *)
12     psi1 = propKin[propPot[Dt/(2M),psi]];
13     psi2 = Nest[propKin[propPot[Dt/M,#]]&, psi1, M-1];
14     propPot[Dt/(2M),psi2]]

```

exercises

Q4.7 Extend the split-step method of section 4.2 to generate not just the final state but all intermediate states as well. *Hint:* use the `NestList` command as in section 4.1.3 (page 79).

Q4.8 Given a particle moving in the range $x \in [0, 1]$ with the scaled non-linear Hamiltonian

$$\hat{\mathcal{H}} = -\frac{1}{\pi^2} \frac{d^2}{dx^2} + \underbrace{\Omega \left[\left(\frac{x - \frac{1}{2}}{\delta} \right)^2 - 1 \right]^2}_{W(x)} + g|\psi(x)|^2, \quad (4.43)$$

do the following calculations:

1. Plot the potential for $\Omega = 1$ and $\delta = \frac{1}{4}$ (use $g = 0$). What are the main characteristics of this potential? *Hint:* compute $V(\frac{1}{2})$, $V(\frac{1}{2} \pm \delta)$, $V'(\frac{1}{2} \pm \delta)$.
2. Calculate and plot the time-dependent density $|\psi(x, t)|^2$ for $\Omega = 50$ and $g = 0$, starting from $\psi_0(x) \propto \exp\left[-\left(\frac{x-x_0}{2\sigma}\right)^2\right]$ with $x_0 = 0.2694$ and $\sigma = 0.0554$. Calculate the probabilities for finding the particle in the left half ($x < \frac{1}{2}$) and in the right half ($x > \frac{1}{2}$) up to $t = 100$. What do you observe?
3. What do you observe for $\Omega = 50$ and $g = 0.1$? Why?

4.2.1 imaginary-time propagation for finding the ground state of the non-linear Schrödinger equation

You may remember from statistical mechanics that at temperature T , the density matrix of a system governed by a Hamiltonian $\hat{\mathcal{H}}$ is

$$\hat{\rho}(\beta) = Z^{-1}(\beta) e^{-\beta \hat{\mathcal{H}}} \quad (4.44)$$

with $\beta = 1/(k_B T)$ in terms of the Boltzmann constant $k_B = 1.3806488(13) \times 10^{-23}$ J/K. The partition function $Z(\beta) = \text{Tr } e^{-\beta \hat{\mathcal{H}}}$ makes sure that the density matrix has the correct norm, $\text{Tr } \hat{\rho} = 1$.

We know that at zero temperature the system will be in its ground state $|\gamma\rangle$,³

$$\lim_{\beta \rightarrow \infty} \hat{\rho}(\beta) = |\gamma\rangle\langle\gamma|. \quad (4.45)$$

If we multiply this equation with an arbitrary state $|\psi\rangle$ from the right, and assume that $\langle\gamma|\psi\rangle \neq 0$, we find

$$\lim_{\beta \rightarrow \infty} \hat{\rho}(\beta)|\psi\rangle = |\gamma\rangle\langle\gamma|\psi\rangle. \quad (4.46)$$

³For simplicity we assume here that the ground state is non-degenerate.

The ground state is therefore

$$|\gamma\rangle = \frac{\lim_{\beta \rightarrow \infty} \hat{\rho}(\beta)|\psi\rangle}{\langle\gamma|\psi\rangle} = \frac{1}{\langle\gamma|\psi\rangle Z(\beta)} \times \lim_{\beta \rightarrow \infty} e^{-\beta \hat{\mathcal{H}}} |\psi\rangle. \quad (4.47)$$

This means that if we take (almost) any state $|\psi\rangle$ and calculate $\lim_{\beta \rightarrow \infty} e^{-\beta \hat{\mathcal{H}}} |\psi\rangle$, we find a state that is proportional to the ground state. But we already know how to do this: the wavefunction $e^{-\beta \hat{\mathcal{H}}} |\psi\rangle$ is calculated from $|\psi\rangle$ by *imaginary-time propagation*. In fact the algorithm of section 4.1.3 remains valid if we replace $i(t - t_0)/\hbar \rightarrow \beta$, and so does its extension to the non-linear Schrödinger equation (section 4.2). The only caveat is that, while regular time propagation (section 4.1.3) is unitary, imaginary-time propagation is not. The wavefunction must therefore be re-normalized after each imaginary-time evolution step (with the `Normalize` function), particularly before calculating the non-linear potential in the Gross–Pitaevskii equation.

For a computer implementation we modify Eq. (4.47) to

$$\begin{aligned} |\gamma\rangle &\propto \lim_{M \rightarrow \infty} e^{-M\delta\beta \hat{\mathcal{H}}} |\psi\rangle = \lim_{M \rightarrow \infty} \left[e^{-\delta\beta \hat{\mathcal{H}}} \right]^M |\psi\rangle \\ &\stackrel{\text{Trotter (4.31)}}{=} \lim_{M \rightarrow \infty} \left[e^{-\frac{\delta\beta}{2} \hat{\mathcal{H}}_{\text{pot}}} e^{-\delta\beta \hat{\mathcal{H}}_{\text{kin}}} e^{-\frac{\delta\beta}{2} \hat{\mathcal{H}}_{\text{pot}}} \right]^M |\psi\rangle \\ &= \lim_{M \rightarrow \infty} e^{-\frac{\delta\beta}{2} \hat{\mathcal{H}}_{\text{pot}}} \underbrace{e^{-\delta\beta \hat{\mathcal{H}}_{\text{kin}}} e^{-\delta\beta \hat{\mathcal{H}}_{\text{pot}}} \dots e^{-\delta\beta \hat{\mathcal{H}}_{\text{kin}}} e^{-\delta\beta \hat{\mathcal{H}}_{\text{pot}}}}_{(M-1) \text{ repetitions of } e^{-\delta\beta \hat{\mathcal{H}}_{\text{kin}}} e^{-\delta\beta \hat{\mathcal{H}}_{\text{pot}}}} e^{-\delta\beta \hat{\mathcal{H}}_{\text{kin}}} e^{-\frac{\delta\beta}{2} \hat{\mathcal{H}}_{\text{pot}}} |\psi\rangle \\ &= e^{-\frac{\delta\beta}{2} \hat{\mathcal{H}}_{\text{pot}}} \left[\lim_{M \rightarrow \infty} \left(e^{-\delta\beta \hat{\mathcal{H}}_{\text{kin}}} e^{-\delta\beta \hat{\mathcal{H}}_{\text{pot}}} \right)^{M-1} \right] e^{-\delta\beta \hat{\mathcal{H}}_{\text{kin}}} e^{-\frac{\delta\beta}{2} \hat{\mathcal{H}}_{\text{pot}}} |\psi\rangle \quad (4.48) \end{aligned}$$

for a fixed “imaginary-time” step $\delta\beta$, and iterate until the term in the square bracket no longer changes and the infinite- β limit ($M \rightarrow \infty$) has effectively been reached.

```

1 In[288]:=groundstate[db_?NumericQ, g_?NumericQ,
2           tolerance_:10^(-10)] :=
3   Module[{Ke,psi0,psi1,psi2,propKin,propPot,gamma},
4     (* compute exponentials *)
5     Ke = Exp[-db*Table[n^2,{n,1,nmax}]];
6     (* propagate with the potential operator *)
7     propPot[ddb_,p_] :=
8       Normalize[Exp[-ddb*(Wval+g*(nmax+1)*Abs[p]^2)] * p];
9     (* propagate with the kinetic operator *)
10    propKin[p_] :=
11      Normalize[FourierDST[Ke*FourierDST[p,1],1]];
12    (* random starting point *)
13    psi0 = Normalize @ RandomComplex[{-1-I,1+I},nmax];
14    (* propagation *)
15    psi1 = propKin[propPot[db/2, Normalize[psi0]]];
16    psi2 = FixedPoint[propKin[propPot[db,#]]&, psi1,
17      SameTest->Function[{p1,p2},Norm[p1-p2]<tolerance]];
18    (* ground state *)
19    gamma = propPot[db/2,psi2];
20    gamma]
```

The last argument, `tolerance`, is optional and is given the value 10^{-10} if not specified. The `FixedPoint` function is used to apply the imaginary-time propagation until the result no longer changes (two consecutive results are considered equal if the function given as `SameTest` returns a true result when applied to these two results).

The ground state of the time-independent non-linear Schrödinger equation satisfies

$$\left[-\frac{\hbar^2}{2m} \frac{d^2}{dx^2} + V(x) + g|\psi(x)|^2 \right] \psi(x) = \mu\psi(x), \quad (4.49)$$

where μ is called the *chemical potential* and takes the place of the ground-state energy in the time-independent *linear* Schrödinger equation. Integrating Eq. (4.49) by $\psi^*(x)$ from the left and integrating over x gives

$$\begin{aligned} \mu &= \int \psi^*(x) \left[-\frac{\hbar^2}{2m} \frac{d^2}{dx^2} + V(x) + g|\psi(x)|^2 \right] \psi(x) dx \\ &= \frac{\hbar^2}{2m} \int \left| \frac{d\psi(x)}{dx} \right|^2 dx + \int [V(x) + g|\psi(x)|^2] |\psi(x)|^2 dx, \end{aligned} \quad (4.50)$$

where we have assumed that $\int |\psi(x)|^2 dx = 1$. We use this to calculate the chemical potential in [In \[288\]](#) by replacing line 20 with

```

20      (* chemical potential *)
21      mu = Table[n^2, {n, 1, nmax}].Abs[FourierDST[gamma, 1]]^2
22          + (Wval + g*(nmax+1)*Abs[gamma]^2).Abs[gamma]^2;
23      (* return ground state and chemical potential *)
24      {mu, gamma}]

```

and adding the local variable `mu` in line 3.

exercises

Q4.9 Given a particle moving in the range $x \in [0, 1]$ with the scaled non-linear Hamiltonian

$$\hat{\mathcal{H}} = -\frac{1}{\pi^2} \frac{d^2}{dx^2} + 500 \left(x - \frac{1}{2} \right)^2 + g|\psi(x)|^2, \quad (4.51)$$

do the following calculations:

1. For $g = 0$ calculate the exact ground state $|\zeta\rangle$ (assuming that the particle can move in $x \in \mathbb{R}$) and its energy eigenvalue. *Hint:* assume $\zeta(x) = \exp \left[-\left(\frac{x - \frac{1}{2}}{2\sigma} \right)^2 \right] / \sqrt{\sigma\sqrt{2\pi}}$ and find the value of σ which minimizes $\langle \zeta | \hat{\mathcal{H}} | \zeta \rangle$.
2. Calculate the ground state $\lim_{\beta \rightarrow \infty} e^{-\beta \hat{\mathcal{H}}} |\zeta\rangle$ and its chemical potential by imaginary-time propagation (with normalization of the wavefunction after each propagation step), using the code given above.
3. Plot the ground-state wavefunction for different values of g .
4. Plot the chemical potential as a function of g .

4.3 several particles in one dimension: interactions

We have seen in section 2.4.2 (page 38) how to describe quantum-mechanical systems with more than one degree of freedom. This method can be used for describing several particles moving in one dimension. In the following we look at two examples of interacting particles.

4.3.1 two particles in one dimension with contact interaction

We first look at two particles moving in a one-dimensional square well of width a and interacting through a contact potential $\delta(x_1 - x_2)$. Such potentials are a good approximation of the interactions taking place in cold dilute gases. The Hamiltonian of this system is

$$\hat{\mathcal{H}} = \underbrace{-\frac{\hbar^2}{2m} \left[\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} \right]}_{\hat{\mathcal{H}}_{\text{kin}}} + \underbrace{V(x_1) + V(x_2)}_{\hat{\mathcal{H}}_{\text{pot}}} + \underbrace{g\delta(x_1 - x_2)}_{\hat{\mathcal{H}}_{\text{int}}}, \quad (4.52)$$

where $V(x)$ is the single-particle potential (as in section 4.1) and $g = \frac{4\pi\hbar^2 a_s}{m}$ is the interaction strength, given through the s -wave scattering length a_s .

We describe this system with the tensor-product basis constructed from two finite-resolution position basis sets:

$$|j_1, j_2\rangle = |j_1\rangle \otimes |j_2\rangle \quad \text{for } j_1, j_2 \in \{1, 2, 3, \dots, n_{\text{max}}\}. \quad (4.53)$$

Most of the matrix representations of the terms in Eq. (4.52) are constructed as tensor products of the matrix representations of the corresponding single-particle representations since $\hat{\mathcal{H}}_{\text{kin}} = \hat{\mathcal{H}}_{\text{kin},1} \otimes \mathbb{1} + \mathbb{1} \otimes \hat{\mathcal{H}}_{\text{kin},2}$ and $\hat{\mathcal{H}}_{\text{pot}} = \hat{\mathcal{H}}_{\text{pot},1} \otimes \mathbb{1} + \mathbb{1} \otimes \hat{\mathcal{H}}_{\text{pot},2}$. The only new element is the interaction Hamiltonian $\hat{\mathcal{H}}_{\text{int}}$. Remembering that its formal operator definition is

$$\hat{\mathcal{H}}_{\text{int}} = g \int_0^a \left[|x_1\rangle \otimes |x_2\rangle \right] \delta(x_1 - x_2) \left[\langle x_1| \otimes \langle x_2| \right] dx_1 dx_2 \quad (4.54)$$

(while Eq. (4.52) is merely a shorthand notation), we calculate its matrix elements as

$$\begin{aligned} \langle j_1, j_2 | \hat{\mathcal{H}}_{\text{int}} | j'_1, j'_2 \rangle &= g \int_0^a \langle j_1 | x_1 \rangle \langle j_2 | x_2 \rangle \delta(x_1 - x_2) \langle x_1 | j'_1 \rangle \langle x_2 | j'_2 \rangle dx_1 dx_2 \\ &= g \int_0^a \vartheta_{j_1}(x) \vartheta_{j_2}(x) \vartheta_{j'_1}(x) \vartheta_{j'_2}(x) dx. \end{aligned} \quad (4.55)$$

We could in principle evaluate these integrals exactly, but notice that there are $\mathcal{O}(n_{\text{max}}^4)$ integrals to be computed, which quickly becomes unmanageably slow as n_{max} grows. Instead, we can again do an approximation: since the basis functions $\vartheta_j(x)$ are zero on all grid points except at x_j [see Eq. (4.10)], the integrand in (4.55) vanishes on all grid points $x_1, x_2, \dots, x_{n_{\text{max}}}$ unless $j_1 = j_2 = j'_1 = j'_2$. We thus approximate the integral by zero if the j -values are not all equal:

$$\langle j_1, j_2 | \hat{\mathcal{H}}_{\text{int}} | j'_1, j'_2 \rangle \approx \delta_{j_1, j_2, j'_1, j'_2} \times g \int_0^a \vartheta_{j_1}^4(x) dx. \quad (4.56)$$

These integrals are not easy to do in all generality. Exact integration of the case $j = \frac{n_{\max}+1}{2}$, which is localized at the center of the square well ($x \approx \frac{1}{2}$, if n_{\max} is odd) gives

$$\int_0^a \partial^4_{\frac{n_{\max}+1}{2}}(x) dx = \frac{1}{3a} \left[2(n_{\max} + 1) + \frac{1}{n_{\max} + 1} \right]. \quad (4.57)$$

We will use this expression to approximate all quartic overlap integrals $\int_0^a \partial_j^4(x) dx$.

Mathematica code: we assume $a = 1$, express energies in units of $E_1 = \frac{\pi^2 \hbar^2}{2ma^2}$, and assume that the potential function $W(x)$ is defined. First we define the grid size and calculate the matrix **X** which converts between the position basis and the momentum basis, as well as the unit operator **id** acting on a single particle:

```

1 In[289]:=nmax = 10;
2 In[290]:=SparseIdentityMatrix[n_] :=
3   SparseArray[Band[{1,1}] -> 1, {n,n}];
4 In[291]:=id = SparseIdentityMatrix[nmax];
5 In[292]:=X = FourierDST[#,1]& /@ SparseIdentityMatrix[nmax];

```

The total kinetic Hamiltonian is assembled via a Kronecker product (tensor product) of the two single-particle kinetic Hamiltonians:

```

6 In[293]:=Hkin1M = SparseArray[Band[{1,1}] -> Range[nmax]^2];
7 In[294]:=Hkin1P = X . Hkin1M . X;
8 In[295]:=HkinP = KroneckerProduct[Hkin1P, id]
9   + KroneckerProduct[id, Hkin1P];

```

The same for the potential Hamiltonian:

```

10 In[296]:=xval = Range[nmax]/(nmax+1);
11 In[297]:=Wval = W /@ xval;
12 In[298]:=Hpot1P = SparseArray[Band[{1,1}] -> Wval];
13 In[299]:=HpotP = KroneckerProduct[Hpot1P, id]
14   + KroneckerProduct[id, Hpot1P];

```

The interaction Hamiltonian is only nonzero when $j_1 = j_2 = j'_1 = j'_2$, which can be represented with a `SparseArray[{j-, j-, j-, j-} -> 1, {nmax, nmax, nmax, nmax}]` massaged into the correct form:

```

15 In[300]:=HintP = (2(nmax+1)+1/(nmax+1))/3 *
16   SparseArray[Flatten /@
17     Flatten[SparseArray[{j-, j-, j-, j-} -> 1,
18       {nmax, nmax, nmax, nmax}], 1]]

```

The full Hamiltonian, in which the amplitude of the potential can be adjusted with the prefactor Ω , is

```

19 In[301]:=HP[Omega_, g_] = HkinP + Omega*HpotP + g*HintP;

```

We calculate eigenstates (the ground state, for example) with the methods already described previously. The resulting wavefunctions are in the tensor-product basis (4.53), and their corresponding densities can be plotted with

```

1 In[302]:=plotdensity[r_] := Module[{r1,r2},
2     (* make square array of density values *)
3     r1 = Partition[r, nmax];
4     (* add zeros at the limits *)
5     r2 = SparseArray[{}, {nmax+2, nmax+2}];
6     r2[[2;;nmax+1, 2;;nmax+1]] = r1;
7     (* plot *)
8     ListDensityPlot[r2, DataRange -> {{0, 1}, {0, 1}}]]

```

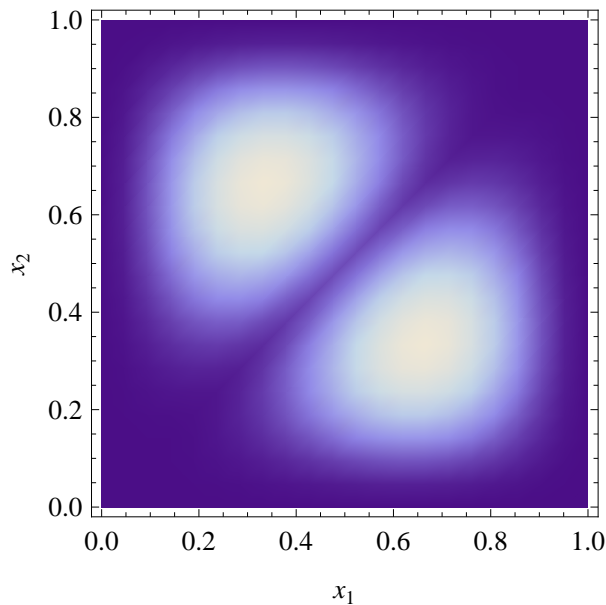
We thus plot a given wavefunction `psi` with

```

1 In[303]:=plotdensity[(nmax+1) * Abs[psi]^2]

```

Here we plot the ground-state density for $\Omega = 0$ (no potential, the particles move in a simple infinite square well) and $g = +5$ (repulsive interaction), using $n_{\max} = 20$ grid points:



We can see that the particles avoid each other, i.e., the density $\rho(x_1, x_2)$ vanishes whenever $x_1 = x_2$.

exercises

Q4.10 Calculate the expectation value of the inter-particle distance, $\langle x_1 - x_2 \rangle$, and its variance, $\langle (x_1 - x_2)^2 \rangle - \langle x_1 - x_2 \rangle^2$, in the ground state as a function of g (still keeping $\Omega = 0$). *Hint:* The position operators `x1` and `x2` are

```

1 In[304]:= x = SparseArray[Band[{1,1}]->xval];
2 In[305]:= x1 = KroneckerProduct[x, id];
3 In[306]:= x2 = KroneckerProduct[id, x];

```

4.3.2 two particles in one dimension with arbitrary interaction

Two particles in one dimension interacting via an arbitrary potential have a Hamiltonian very similar to Eq. (4.52), except that the interaction is now

$$\hat{\mathcal{H}}_{\text{int}} = V_{\text{int}}(x_1, x_2). \quad (4.58)$$

As an example, for the Coulomb interaction we have $V_{\text{int}}(x_1, x_2) = \frac{Q_1 Q_2}{4\pi\epsilon_0 |x_1 - x_2|}$ with Q_1 and Q_2 the electric charges of the two particles. For many realistic potentials V_{int} only depends on $|x_1 - x_2|$.

The matrix elements of this interaction Hamiltonian can be approximated with a method similar to what we have already seen. The exact expression

$$\langle j_1, j_2 | \hat{\mathcal{H}}_{\text{int}} | j'_1, j'_2 \rangle = \int_0^a \vartheta_{j_1}(x_1) \vartheta_{j_2}(x_2) V_{\text{int}}(x_1, x_2) \vartheta_{j'_1}(x_1) \vartheta_{j'_2}(x_2) dx_1 dx_2 \quad (4.59)$$

is approximated by noticing that on the grid points, the numerator of the integrand is only nonzero if $j_1 = j'_1$ and $j_2 = j'_2$:

$$\langle j_1, j_2 | \hat{\mathcal{H}}_{\text{int}} | j'_1, j'_2 \rangle \approx \delta_{j_1, j'_1} \delta_{j_2, j'_2} \times \int_0^a \vartheta_{j_1}^2(x_1) \vartheta_{j_2}^2(x_2) V_{\text{int}}(x_1, x_2) dx_1 dx_2. \quad (4.60)$$

With $\vartheta_j(x) \approx \delta(x - x_j)$ (in fact it is the best approximation to a Dirac δ -function possible in our finite-resolution basis), these matrix elements simplify to

$$\begin{aligned} \langle j_1, j_2 | \hat{\mathcal{H}}_{\text{int}} | j'_1, j'_2 \rangle &\approx \delta_{j_1, j'_1} \delta_{j_2, j'_2} \times \int_0^a \delta(x_1 - x_{j_1}) \delta(x_2 - x_{j_2}) V_{\text{int}}(x_1, x_2) dx_1 dx_2 \\ &= \delta_{j_1, j'_1} \delta_{j_2, j'_2} \times V_{\text{int}}(x_{j_1}, x_{j_2}). \end{aligned} \quad (4.61)$$

This is again very easy to evaluate without the need for integration over basis functions. But realistic interaction potentials are usually singular for $x_1 = x_2$ (consider, for example, the Coulomb potential), and therefore this expression (4.61) fails for the evaluation of the matrix elements $\langle j, j | \hat{\mathcal{H}}_{\text{int}} | j, j \rangle$. This problem cannot be solved in all generality, and we can either resort to more accurate integration (as in section 4.3.1) or we can replace the true interaction potential with a less singular version: for the Coulomb potential, we could for example use a truncated singularity for $|x| < \Delta$:

$$V_{\text{int}}(x) = \frac{Q_1 Q_2}{4\pi\epsilon_0} \times \begin{cases} \frac{1}{|x|} & \text{if } |x| \geq \Delta \\ \frac{3\Delta^2 - x^2}{2\Delta^3} & \text{if } |x| < \Delta \end{cases} \quad (4.62)$$

As long as the particles move at energies much smaller than $V_{\text{int}}(\pm\Delta) = \frac{Q_1 Q_2}{4\pi\epsilon_0 \Delta}$ they cannot distinguish the true Coulomb potential from this truncated form.

exercises

- Q4.11** Consider two particles in an infinite square well, interacting via the truncated Coulomb potential (4.62). Calculate the expectation value of the inter-particle distance, $\langle x_1 - x_2 \rangle$, and its variance, $\langle (x_1 - x_2)^2 \rangle - \langle x_1 - x_2 \rangle^2$, in the ground state as a function of the Coulomb interaction strength (attractive and repulsive). *Hint:* set $\Delta = a/(n_{\text{max}} + 1)$ in Eq. (4.62).

4.4 one particle in several dimensions

An important application of the imaginary-time propagation method of section 4.2.1 is the calculation of the shape of a three-dimensional Bose–Einstein condensate in a harmonic trap. In this section we use such a calculation as an example of how to extend single-particle lattice quantum mechanics to more spatial dimensions.

The non-linear Hamiltonian describing a three-dimensional Bose–Einstein condensate in a harmonic trap is

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) + \frac{m}{2} \left(\omega_x^2 x^2 + \omega_y^2 y^2 + \omega_z^2 z^2 \right) + (N-1) \frac{4\pi\hbar^2 a_s}{m} |\psi(x, y, z)|^2, \quad (4.63)$$

where we have assumed that the single-particle wavefunction $\psi(x, y, z)$ is normalized: $\int |\psi(x, y, z)|^2 dx dy dz = 1$.

We perform this calculation in a square box, where $|x| \leq \frac{a}{2}$, $|y| \leq \frac{a}{2}$, and $|z| \leq \frac{a}{2}$; we will need to choose a large enough so that the BEC fits into this box, but small enough so that we do not need an unreasonably large n_{\max} for the description of its wavefunction. Notice that this box is shifted by $\frac{a}{2}$ compared to the $[0 \dots a]$ boxes used so far; this does not influence the calculations in any way. The energy scale of this box is $E_1 = \frac{\pi^2 \hbar^2}{2ma^2}$. Introducing the dimensionless coordinates $\tilde{x} = x/a$, $\tilde{y} = y/a$, and $\tilde{z} = z/a$, and defining the dimensionless wavefunction $\tilde{\psi}(\tilde{x}, \tilde{y}, \tilde{z}) = a^{3/2} \psi(x, y, z)$, the dimensionless Hamiltonian is found from Eq. (4.63):

$$\frac{\hat{\mathcal{H}}}{E_1} = -\frac{1}{\pi^2} \left(\frac{\partial^2}{\partial \tilde{x}^2} + \frac{\partial^2}{\partial \tilde{y}^2} + \frac{\partial^2}{\partial \tilde{z}^2} \right) + \Omega_x^2 \tilde{x}^2 + \Omega_y^2 \tilde{y}^2 + \Omega_z^2 \tilde{z}^2 + (N-1) \gamma |\tilde{\psi}(\tilde{x}, \tilde{y}, \tilde{z})|^2, \quad (4.64)$$

where $\Omega_x = \frac{m\omega_x a^2}{\pi\hbar}$ etc. are the dimensionless trap frequencies and $\gamma = \frac{8a_s}{\pi a}$ is the dimensionless scattering length (interaction strength).

The ground state of this dimensionless non-linear Hamiltonian (4.64) can be found by three-dimensional imaginary-time propagation, starting from (almost) any arbitrary state. Here we assemble a Mathematica function `groundstate` which, given an initial state `psi0` and an imaginary time step `db`, propagates until the state is converged to the ground state.

First we define the dimensionless parameters of the problem. We will be considering $N = 1000$ ^{87}Rb atoms in a magnetic trap with trap frequencies $\omega_x = 2\pi \times 115 \text{ Hz}$ and $\omega_y = \omega_z = 2\pi \times 540 \text{ Hz}$. The ^{87}Rb atoms are assumed to be in the $|F=1, M_F=1\rangle$ hyperfine ground state, where their s -wave scattering length is $a_s = 100.4 a_0$ (with $a_0 = 52.9177 \text{ pm}$ the Bohr radius).

```

1 In[307]:=With[{m = Quantity["86.909187 u"],
2           a = Quantity["10 um"],
3           wx = 2 Pi Quantity["115 Hz"],
4           wy = 2 Pi Quantity["540 Hz"],
5           wz = 2 Pi Quantity["540 Hz"],
6           as = Quantity["100.4 a0"],
7           h = Quantity["h"]},
8   Ox = m wx a^2/(Pi h);
9   Oy = m wy a^2/(Pi h);
10  Oz = m wz a^2/(Pi h);
11  g = 8 as/(Pi a);]
```


Next we define the grid on which the calculations will be done. In each Cartesian direction there are n_{\max} grid points $\tilde{x}_j = \text{xval}[[j]]$:

```
12 In[308]:=nmax = 41;
13 In[309]:=xval = Range[nmax]/(nmax+1) - 1/2;
```

We define the dimensionless harmonic-trap potential: the potential has its minimum at the center of the calculation box, i.e., at $\tilde{x} = \tilde{y} = \tilde{z} = \frac{1}{2}$.

```
14 In[310]:=W[x_,y_,z_] = 0x^2*x^2 + 0y^2*y^2 + 0z^2*z^2;
```

We only need the values of this potential on the grid points. To evaluate this, we build a three-dimensional array whose element $\text{Wval}[[jx,jy,jz]]$ is given by the grid-point value $W[\text{xval}[[jx]],\text{xval}[[jy]],\text{xval}[[jz]]$:

```
15 In[311]:=Wval = Table[W[xval[[jx]],xval[[jy]],xval[[jz]]],
16               {jx,nmax}, {jy,nmax}, {jz,nmax};
```

We could also define this more efficiently through functional programming:

```
17 In[312]:=Wval = Outer[W, xval, xval, xval];
```

The structure of the three-dimensional Wval array of potential values mirrors the structure of the wavefunction that we will be using: any wavefunction psi will be a $n_{\max} \times n_{\max} \times n_{\max}$ array of coefficients in our finite-resolution position basis:

$$\tilde{\psi}(\tilde{x}, \tilde{y}, \tilde{z}) = \sum_{j_x, j_y, j_z=1}^{n_{\max}} \text{psi}[[jx,jy,jz]] \theta_{j_x}(\tilde{x}) \theta_{j_y}(\tilde{y}) \theta_{j_z}(\tilde{z}). \quad (4.65)$$

From Eq. (4.10) we find that on the three-dimensional grid points the wavefunction takes the values

$$\tilde{\psi}(\tilde{x}_{j_x}, \tilde{x}_{j_y}, \tilde{x}_{j_z}) = (n_{\max} + 1)^{3/2} \text{psi}[[jx,jy,jz]]. \quad (4.66)$$

The norm of a wavefunction is

$$\begin{aligned} \int_0^a |\psi(x, y, z)|^2 dx dy dz &= \int_0^1 |\tilde{\psi}(\tilde{x}, \tilde{y}, \tilde{z})|^2 d\tilde{x} d\tilde{y} d\tilde{z} = \sum_{j_x, j_y, j_z=1}^{n_{\max}} |\text{psi}[[jx,jy,jz]]|^2 \\ &= \text{Norm}[\text{Flatten}[\text{psi}]]^2, \end{aligned} \quad (4.67)$$

from which we define a wavefunction normalization function

```
18 In[313]:=nn[psi_] := psi/Norm[Flatten[psi]]
```

The ground state calculation then goes by imaginary-time propagation, with step size db corresponding to an evolution $e^{-\text{db} \hat{\mathcal{H}}/E_1}$ per step. The calculation is done for $N = n$ particles. Notice that the `FourierDST` function can do multi-dimensional discrete sine transforms, and therefore the kinetic-energy propagator can still be evaluated very efficiently. The last argument, `tolerance`, is optional and is given the value 10^{-6} if not specified.

```

19 In[314]:=groundstate[n_?NumericQ, db_?NumericQ,
20   tolerance_:10^(-6)] :=
21   Module[{Ke,propKin,propPot,psi0,psi1,psi2,gamma,mu},
22     (* kinetic propagator in momentum basis *)
23     Ke = Table[Exp[-db*(nx^2+ny^2+nz^2)],
24       {nx,nmax}, {ny,nmax}, {nz,nmax}] //N;
25     (* kinetic propagator in position basis *)
26     propKin[psi_] := FourierDST[Ke*FourierDST[psi,1],1];
27     (* random starting point *)
28     psi0 = nn@RandomComplex[{-1-I,1+I},{nmax,nmax,nmax}];
29     (* potential propagator in position basis *)
30     propPot[b_?NumericQ, psi_] :=
31       Exp[-b*(Wval+g*(n-1)*(nmax+1)^3*Abs[psi]^2)]*psi;
32     (* first evolution step *)
33     psi1 = nn[propKin[propPot[db/2,nn[psi0]]]];
34     (* iterate evolution until wavefunction converges *)
35     psi2 = FixedPoint[nn[propKin[propPot[db,#]]]&, psi1,
36       SameTest -> (Norm[Flatten[#1-#2]] < tolerance &)];
37     (* one last half-iteration *)
38     gamma = nn[propPot[db/2, psi2]];
39     (* chemical potential *)
40     mu = Flatten[Table[nx^2+ny^2+nz^2,
41       {nx,nmax},{ny,nmax},{nz,nmax}]] .
42       Flatten[Abs[FourierDST[gamma,1]]^2]
43       + Flatten[(Wval+g*(n-1)*(nmax+1)^3*Abs[gamma]^2)] .
44       Flatten[Abs[gamma]^2];
45     (* return ground state and chemical potential *)
46     {mu, gamma}]

```

As an example, we calculate the ground state for $N = 1000$ atoms and a time step of $db = 0.001$:

```

1 In[315]:= {pg,p} = groundstate[1000, 0.001];

```

The chemical potential is

```

1 In[316]:= pg
2 Out[316]= 228.421

```

From this result we can, for example, calculate the expectation values $X = \langle x \rangle$, $Y = \langle y \rangle$, $Z = \langle z \rangle$, $XX = \langle x^2 \rangle$, $YY = \langle y^2 \rangle$, $ZZ = \langle z^2 \rangle$. We could define coordinate arrays as

```

1 In[317]:= xc = Table[xval[[jx]], {jx,nmax}, {jy,nmax}, {jz,nmax}];
2 In[318]:= yc = Table[xval[[jy]], {jx,nmax}, {jy,nmax}, {jz,nmax}];
3 In[319]:= zc = Table[xval[[jz]], {jx,nmax}, {jy,nmax}, {jz,nmax}];

```

or we could define them more efficiently as follows:

```

1 In[320]:= ones = Array[1&, nmax];
2 In[321]:= xc = Outer[Times, xval, ones, ones];

```

```

3 In[322]:=yc = Outer[Times, ones, xval, ones];
4 In[323]:=zc = Outer[Times, ones, ones, xval];

```

The desired expectation values are then computed with

```

1 In[324]:=X = Total[Flatten[xc * Abs[p]^2]];
2 In[325]:=Y = Total[Flatten[yc * Abs[p]^2]];
3 In[326]:=Z = Total[Flatten[zc * Abs[p]^2]];
4 In[327]:=XX = Total[Flatten[xc^2 * Abs[p]^2]];
5 In[328]:=YY = Total[Flatten[yc^2 * Abs[p]^2]];
6 In[329]:=ZZ = Total[Flatten[zc^2 * Abs[p]^2]];

```

The size of the BEC is then calculated from these as the standard deviations of the position in the three Cartesian directions:

```

1 In[330]:={Sqrt[XX-X^2], Sqrt[YY-Y^2], Sqrt[ZZ-Z^2]}
2 Out[330]={0.158723, 0.0419921, 0.0419921}

```

exercises

Q4.12 Take the BEC Hamiltonian (4.63) in the absence of interactions ($a_s = 0$) and calculate analytically the expectation values $\langle x^2 \rangle$, $\langle y^2 \rangle$, $\langle z^2 \rangle$ in the ground state.

Q4.13 Take the BEC Hamiltonian (4.63) in the limit of strong interactions (Thomas–Fermi limit) where the kinetic energy can be neglected. The Gross–Pitaevskii equation is then

$$\left[\frac{m}{2} (\omega_x^2 x^2 + \omega_y^2 y^2 + \omega_z^2 z^2) + (N-1) \frac{4\pi\hbar^2 a_s}{m} |\psi(x, y, z)|^2 \right] \psi(x, y, z) = \mu \psi(x, y, z), \quad (4.68)$$

which has two solutions:

$$|\psi(x, y, z)|^2 = \begin{cases} 0 & \text{or} \\ \frac{\mu - \frac{m}{2} (\omega_x^2 x^2 + \omega_y^2 y^2 + \omega_z^2 z^2)}{(N-1) \frac{4\pi\hbar^2 a_s}{m}} & . \end{cases} \quad (4.69)$$

Together with the conditions that $|\psi(x, y, z)|^2 \geq 0$, that $\psi(x, y, z)$ should be continuous, and that $\int |\psi(x, y, z)|^2 dx dy dz = 1$, this gives us the Thomas–Fermi “inverted parabola” density

$$|\psi(x, y, z)|^2 = \begin{cases} \rho_0 \left[1 - \left(\frac{x}{R_x} \right)^2 - \left(\frac{y}{R_y} \right)^2 - \left(\frac{z}{R_z} \right)^2 \right] & \text{if } \left(\frac{x}{R_x} \right)^2 + \left(\frac{y}{R_y} \right)^2 + \left(\frac{z}{R_z} \right)^2 \leq 1, \\ 0 & \text{if not,} \end{cases} \quad (4.70)$$

with the central density

$$\rho_0 = \frac{1}{8\pi} \left[\frac{225 m^6 \omega_x^2 \omega_y^2 \omega_z^2}{\hbar^6 a_s^3 (N-1)^3} \right]^{\frac{1}{5}}, \quad (4.71)$$

the Thomas–Fermi radii

$$R_x = \left[\frac{15\hbar^2 a_s (N-1) \omega_y \omega_z}{m^2 \omega_x^4} \right]^{\frac{1}{5}}, \quad R_y = \left[\frac{15\hbar^2 a_s (N-1) \omega_z \omega_x}{m^2 \omega_y^4} \right]^{\frac{1}{5}}, \quad R_z = \left[\frac{15\hbar^2 a_s (N-1) \omega_x \omega_y}{m^2 \omega_z^4} \right]^{\frac{1}{5}}, \quad (4.72)$$

and the chemical potential

$$\mu = \frac{1}{2} \left[225 m \hbar^4 a_s^2 (N-1)^2 \omega_x^2 \omega_y^2 \omega_z^2 \right]^{\frac{1}{5}}. \quad (4.73)$$

Using this density, calculate the expectation values $\langle x^2 \rangle$, $\langle y^2 \rangle$, $\langle z^2 \rangle$ in the ground state of the Thomas–Fermi approximation.

- Q4.14** Compare the numerical expectation values $\langle x^2 \rangle$, $\langle y^2 \rangle$, $\langle z^2 \rangle$ of our Mathematica code to the analytic results of Q4.12 and Q4.13. What is the maximum ^{87}Rb atom number N which allows a reasonably good description (in this specific trap) with the non-interacting solution? What is the minimum atom number which allows a reasonably good description with the Thomas–Fermi solution?
- Q4.15** Consider a ^{87}Rb Bose–Einstein condensate in a harmonic trap, described by the non-linear Hamiltonian (4.63). Take $\omega_y = \omega_z = 2\pi \times 500\text{Hz}$ and a scattering length $a_s = 100.4a_0$. Compute the expectation values $\langle x^2 \rangle$, $\langle y^2 \rangle$, $\langle z^2 \rangle$ for several values of ω_x and try to interpret the asymptotes $\omega_x \rightarrow 0$ and $\omega_x \rightarrow \infty$.

Chapter 5

combining space and spin

In this chapter we put many of the techniques studied so far together: spin degrees of freedom (chapter 3) and spatial degrees of freedom (chapter 4) are combined with the tensor-product formalism (chapter 2).

5.1 one particle with spin in one dimension

5.1.1 separable Hamiltonian

The simplest problem combining a spatial and a spin degree of freedom in a meaningful way consists of a single spin-1/2 particle moving in one dimension in a state-selective potential:

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m} \frac{d^2}{dx^2} + V_0(x) + V_z(x) \hat{\sigma}_z. \quad (5.1)$$

As was said before, Eq. (5.1) is a short-hand notation of the full Hamiltonian

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m} \int_{-\infty}^{\infty} dx |x\rangle \frac{d^2}{dx^2} \langle x| \otimes \mathbb{1} + \int_{-\infty}^{\infty} dx |x\rangle V_0(x) \langle x| \otimes \mathbb{1} + \int_{-\infty}^{\infty} dx |x\rangle V_z(x) \langle x| \otimes \sigma_z, \quad (5.2)$$

where it is more evident that the first two terms act only on the spatial part of the wavefunction, while the third term couples the two degrees of freedom.

The Hilbert space of this particle consists of a one-dimensional degree of freedom x , which we had described in chapter 4 with a basis built from square-well eigenstates, and a spin-1/2 degree of freedom $\vec{\sigma}$ described in the Dicke basis (chapter 3). This tensor-product structure of the Hilbert space allows us to simplify the

matrix elements of the Hamiltonian by factoring out the spin degree of freedom,

$$\begin{aligned}
\langle \phi, \uparrow | \hat{\mathcal{H}} | \psi, \uparrow \rangle &= -\frac{\hbar^2}{2m} \int_{-\infty}^{\infty} \phi^*(x) \psi''(x) dx \langle \uparrow | \uparrow \rangle + \int_{-\infty}^{\infty} \phi^*(x) V_0(x) \psi(x) dx \langle \downarrow | \downarrow \rangle + \int_{-\infty}^{\infty} \phi^*(x) V_z(x) \psi(x) dx \langle \uparrow | \hat{\sigma}_z | \uparrow \rangle \\
&= -\frac{\hbar^2}{2m} \int_{-\infty}^{\infty} \phi^*(x) \psi''(x) dx + \int_{-\infty}^{\infty} \phi^*(x) V_0(x) \psi(x) dx + \frac{1}{2} \int_{-\infty}^{\infty} \phi^*(x) V_z(x) \psi(x) dx \\
\langle \phi, \uparrow | \hat{\mathcal{H}} | \psi, \downarrow \rangle &= -\frac{\hbar^2}{2m} \int_{-\infty}^{\infty} \phi^*(x) \psi''(x) dx \langle \uparrow | \downarrow \rangle + \int_{-\infty}^{\infty} \phi^*(x) V_0(x) \psi(x) dx \langle \uparrow | \downarrow \rangle + \int_{-\infty}^{\infty} \phi^*(x) V_z(x) \psi(x) dx \langle \uparrow | \hat{\sigma}_z | \downarrow \rangle \\
&= 0 \\
\langle \phi, \downarrow | \hat{\mathcal{H}} | \psi, \uparrow \rangle &= -\frac{\hbar^2}{2m} \int_{-\infty}^{\infty} \phi^*(x) \psi''(x) dx \langle \downarrow | \uparrow \rangle + \int_{-\infty}^{\infty} \phi^*(x) V_0(x) \psi(x) dx \langle \downarrow | \uparrow \rangle + \int_{-\infty}^{\infty} \phi^*(x) V_z(x) \psi(x) dx \langle \downarrow | \hat{\sigma}_z | \uparrow \rangle \\
&= 0 \\
\langle \phi, \downarrow | \hat{\mathcal{H}} | \psi, \downarrow \rangle &= -\frac{\hbar^2}{2m} \int_{-\infty}^{\infty} \phi^*(x) \psi''(x) dx \langle \downarrow | \downarrow \rangle + \int_{-\infty}^{\infty} \phi^*(x) V_0(x) \psi(x) dx \langle \downarrow | \downarrow \rangle + \int_{-\infty}^{\infty} \phi^*(x) V_z(x) \psi(x) dx \langle \downarrow | \hat{\sigma}_z | \downarrow \rangle \\
&= -\frac{\hbar^2}{2m} \int_{-\infty}^{\infty} \phi^*(x) \psi''(x) dx + \int_{-\infty}^{\infty} \phi^*(x) V_0(x) \psi(x) dx - \frac{1}{2} \int_{-\infty}^{\infty} \phi^*(x) V_z(x) \psi(x) dx.
\end{aligned} \tag{5.3}$$

We see that this Hamiltonian does not mix states with different spin states (since all matrix elements where the spin state differs between the left and right side are equal to zero). We can therefore solve the two disconnected problems of finding the particle's behavior with spin up or with spin down, with effective Hamiltonians

$$\hat{\mathcal{H}}_{\uparrow} = -\frac{\hbar^2}{2m} \frac{d^2}{dx^2} + V_0(x) + \frac{1}{2} V_z(x), \tag{5.4a}$$

$$\hat{\mathcal{H}}_{\downarrow} = -\frac{\hbar^2}{2m} \frac{d^2}{dx^2} + V_0(x) - \frac{1}{2} V_z(x). \tag{5.4b}$$

These Hamiltonians now only describe the spatial degree of freedom, and the methods of chapter 4 can be used without further modifications.

5.1.2 non-separable Hamiltonian

A more interesting situation arises when the Hamiltonian is not separable as in section 5.1.1. Take, for example, the Hamiltonian of Eq. (5.1) in the presence of a transverse magnetic field B_x ,

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m} \frac{d^2}{dx^2} + V_0(x) + V_z(x) \hat{\sigma}_z + B_x \hat{\sigma}_x. \tag{5.5}$$

The interaction Hamiltonian with the magnetic field is not separable:

$$\begin{aligned}
\langle \phi, \uparrow | B_x \hat{\sigma}_x | \psi, \uparrow \rangle &= B_x \int_{-\infty}^{\infty} \phi^*(x) \psi(x) dx \langle \uparrow | \hat{\sigma}_x | \uparrow \rangle = 0 \\
\langle \phi, \uparrow | B_x \hat{\sigma}_x | \psi, \downarrow \rangle &= B_x \int_{-\infty}^{\infty} \phi^*(x) \psi(x) dx \langle \uparrow | \hat{\sigma}_x | \downarrow \rangle = \frac{1}{2} B_x \int_{-\infty}^{\infty} \phi^*(x) \psi(x) dx \\
\langle \phi, \downarrow | B_x \hat{\sigma}_x | \psi, \uparrow \rangle &= B_x \int_{-\infty}^{\infty} \phi^*(x) \psi(x) dx \langle \downarrow | \hat{\sigma}_x | \uparrow \rangle = \frac{1}{2} B_x \int_{-\infty}^{\infty} \phi^*(x) \psi(x) dx \\
\langle \phi, \downarrow | B_x \hat{\sigma}_x | \psi, \downarrow \rangle &= B_x \int_{-\infty}^{\infty} \phi^*(x) \psi(x) dx \langle \downarrow | \hat{\sigma}_x | \downarrow \rangle = 0.
\end{aligned} \tag{5.6}$$

Therefore we can no longer study separate Hamiltonians as in Eqs. (5.4), and we must instead study the joint system of spatial motion and spin. In what follows we study a simple example of such a Hamiltonian, both analytically and numerically. We take the trapping potential to be harmonic,

$$V_0(x) = \frac{1}{2}m\omega^2 x^2 \quad (5.7)$$

and the state-selective potential as a homogeneous force,

$$V_z(x) = -Fx. \quad (5.8)$$

ground state for $B_x = 0$

For $B_x = 0$ we know that the ground states of the two spin sectors are the ground states of the effective Hamiltonians (5.4), which are Gaussians:

$$\langle x|\gamma_\uparrow\rangle = \frac{e^{-\left(\frac{x-\mu}{2\sigma}\right)^2}}{\sqrt{\sigma\sqrt{2\pi}}} \otimes |\uparrow\rangle \quad \langle x|\gamma_\downarrow\rangle = \frac{e^{-\left(\frac{x+\mu}{2\sigma}\right)^2}}{\sqrt{\sigma\sqrt{2\pi}}} \otimes |\downarrow\rangle \quad (5.9)$$

with $\mu = \frac{F}{2m\omega^2}$ and $\sigma = \sqrt{\frac{\hbar}{2m\omega}}$. These two ground states are degenerate, with energy $E = \frac{1}{2}\hbar\omega - \frac{F^2}{8m\omega^2}$. In both of these ground states the spatial and spin degrees of freedom are entangled: the particle is more likely to be detected in the $|\uparrow\rangle$ state on the right side ($x > 0$), and more likely to be detected in the $|\downarrow\rangle$ state on the left side ($x < 0$) of the trap. This results in a positive expectation value of the operator $\hat{x} \otimes \hat{\sigma}_z$:

$$\langle \gamma_\uparrow | \hat{x} \otimes \hat{\sigma}_z | \gamma_\uparrow \rangle = \langle \gamma_\downarrow | \hat{x} \otimes \hat{\sigma}_z | \gamma_\downarrow \rangle = \frac{\mu}{2} = \frac{F}{4m\omega^2}. \quad (5.10)$$

perturbative ground state for $B_x > 0$

For small $|B_x|$ the ground state can be described by a linear combination of the states (5.9). If we set

$$|\gamma_p\rangle = \alpha \times |\gamma_\uparrow\rangle + \beta \times |\gamma_\downarrow\rangle \quad (5.11)$$

with $|\alpha|^2 + |\beta|^2 = 1$, we find that the expectation value of the energy is

$$\begin{aligned} \langle \gamma_p | \hat{\mathcal{H}} | \gamma_p \rangle &= |\alpha|^2 \langle \gamma_\uparrow | \hat{\mathcal{H}} | \gamma_\uparrow \rangle + \alpha^* \beta \langle \gamma_\uparrow | \hat{\mathcal{H}} | \gamma_\downarrow \rangle + \beta^* \alpha \langle \gamma_\downarrow | \hat{\mathcal{H}} | \gamma_\uparrow \rangle + |\beta|^2 \langle \gamma_\downarrow | \hat{\mathcal{H}} | \gamma_\downarrow \rangle \\ &= \frac{1}{2}\hbar\omega - \frac{F^2}{8m\omega^2} + \frac{1}{2}B_x(\alpha^* \beta + \beta^* \alpha) e^{-\frac{F^2}{4m\hbar\omega^3}} \end{aligned} \quad (5.12)$$

For $B_x > 0$ this energy is minimized for $\alpha = 1/\sqrt{2}$ and $\beta = -1/\sqrt{2}$, and the perturbative ground state is therefore the anti-symmetric combination of the states (5.9)

$$\langle x|\gamma_p\rangle = \frac{e^{-\left(\frac{x-\mu}{2\sigma}\right)^2}}{\sqrt{2\sigma\sqrt{2\pi}}} \otimes |\uparrow\rangle - \frac{e^{-\left(\frac{x+\mu}{2\sigma}\right)^2}}{\sqrt{2\sigma\sqrt{2\pi}}} \otimes |\downarrow\rangle. \quad (5.13)$$

with energy

$$\langle \gamma_p | \hat{\mathcal{H}} | \gamma_p \rangle = \frac{1}{2}\hbar\omega - \frac{F^2}{8m\omega^2} - \frac{1}{2}B_x e^{-\frac{F^2}{4m\hbar\omega^3}}. \quad (5.14)$$

The energy splitting between this ground state and the first excited state,

$$\langle x | \epsilon_p \rangle = \frac{e^{-\left(\frac{x-\mu}{2\sigma}\right)^2}}{\sqrt{2\sigma}\sqrt{2\pi}} \otimes |\uparrow\rangle + \frac{e^{-\left(\frac{x+\mu}{2\sigma}\right)^2}}{\sqrt{2\sigma}\sqrt{2\pi}} \otimes |\downarrow\rangle. \quad (5.15)$$

is $\Delta E = \langle \epsilon_p | \hat{\mathcal{H}} | \epsilon_p \rangle - \langle \gamma_p | \hat{\mathcal{H}} | \gamma_p \rangle = B_x e^{-\frac{F^2}{4m\hbar\omega^3}}$, which can be very small for large exponents $\frac{F^2}{4m\hbar\omega^3}$.

numerical calculation of the ground state

For a numerical description of this particle we first re-scale the Hamiltonian to eliminate unnecessary units. As usual we describe the spatial degree of freedom in a box of size a , with energy scale $E_1 = \frac{\pi^2 \hbar^2}{2ma^2}$; we set $x = a\tilde{x}$ and use the range $-\frac{1}{2} < \tilde{x} < \frac{1}{2}$.¹ The scaled Hamiltonian is

$$\frac{\hat{\mathcal{H}}}{E_1} = -\frac{1}{\pi^2} \frac{d^2}{d\tilde{x}^2} + \Omega^2 \hat{x}^2 - f \hat{x} \otimes \hat{\sigma}_z + b_x \hat{\sigma}_x \quad (5.16)$$

with $\Omega = \omega \frac{ma^2}{\pi \hbar}$, $f = F \frac{2ma^3}{\pi^2 \hbar^2}$, and $b_x = B_x \frac{2ma^2}{\pi^2 \hbar^2}$ the dimensionless parameters of the problem.

We describe the spatial degree of freedom with the finite-resolution position basis of section 4.1.1:

```
1 In[331]:=nmax = 20;
2 In[332]:=xval = Range[nmax]/(nmax+1)-1/2;
```

The operator \hat{x} is approximately diagonal in this representation:

```
3 In[333]:=xop = SparseArray[Band[{1,1}] -> xval];
```

The identity operator on the spatial degree of freedom is

```
4 In[334]:=idx = SparseArray[Band[{1,1}] -> 1, {nmax,nmax}];
```

The Pauli operators for the spin degree of freedom are

```
5 In[335]:=sx = {{0,1},{1,0}}/2 //SparseArray;
6 In[336]:=sy = {{0,-I},{I,0}}/2 //SparseArray;
7 In[337]:=sz = {{1,0},{0,-1}}/2 //SparseArray;
8 In[338]:=ids = {{1,0},{0,1}} //SparseArray;
```

The kinetic energy operator is constructed via a discrete sine transform, as before:

```
9 In[339]:=HkinM = SparseArray[{n_,n_} -> n^2, {nmax, nmax}];
10 In[340]:=X = FourierDST[#,1]& /@ idx;
11 In[341]:=HkinP = X . HkinM . X;
```

¹Until now we had always used $0 < \tilde{x} < 1$. Shifting this domain to $-\frac{1}{2} < \tilde{x} < \frac{1}{2}$ does not change anything in the computational methods presented so far.

From these we assemble the Hamiltonian:

```

12 In[342]:=H[Omega_, f_, bx_] =
13 In[343]:= KroneckerProduct[HkinP, ids]
14 In[344]:= + Omega^2 * KroneckerProduct[xop.xop, ids]
15 In[345]:= - f * KroneckerProduct[xop, sz]
16 In[346]:= + bx * KroneckerProduct[idx, sx];

```

We compute the ground state of this Hamiltonian with

```

17 In[347]:=Clear[gs];
18 In[348]:=gs[Omega_?NumericQ, f_?NumericQ, bx_?NumericQ] :=
19     gs[Omega, f, bx] =
20     -Eigensystem[-H[N[Omega],N[f],N[bx]], 1,
21         Method -> {"Arnoldi", "Criteria" -> "RealPart",
22             MaxIterations -> 10^6}]

```

Once a ground state $|\gamma\rangle$ has been calculated, for example with

```

1 In[349]:=gamma = gs[100, 10000, 1000][[2, 1]];
2 In[350]:=Dimensions[gamma]
3 Out[350]={40}

```

the usual problem arises of how to display and interpret the wavefunction. In order to facilitate this analysis, we first re-shape the ground state to better reflect the tensor-product structure of our Hilbert space:

```

1 In[351]:=gammaA = Partition[gamma, 2];
2 In[352]:=Dimensions[gammaA]
3 Out[352]={20, 2}

```

In this way, `gammaA[[j,s]]` is the coefficient corresponding to the basis function $|j\rangle \otimes |\frac{3}{2} - s\rangle = |j, \frac{3}{2} - s\rangle$, with the definitions $|+\frac{1}{2}\rangle = |\uparrow\rangle$ and $|-\frac{1}{2}\rangle = |\downarrow\rangle$ for the spin part (so that $s = 1$ corresponds to the $|\uparrow\rangle$ and $s = 2$ to the $|\downarrow\rangle$ state). From this re-shaped ground state we calculate the re-shaped density matrix

```

1 In[353]:=rhoA = Outer[Times, gammaA, Conjugate[gammaA]];
2 In[354]:=Dimensions[rhoA]
3 Out[354]={20, 2, 20, 2}

```

In this density matrix, $c_{j_1, s_1, j_2, s_2} = \text{rhoA}[[j_1, s_1, j_2, s_2]]$ is the coefficient corresponding to the basis function $|j_1, \frac{3}{2} - s_1\rangle \langle j_2, \frac{3}{2} - s_2|$ in the basis expansion of the density matrix:

$$\hat{\rho} = \sum_{j_1=1}^{n_{\max}} \sum_{s_1=1}^2 \sum_{j_2=1}^{n_{\max}} \sum_{s_2=1}^2 c_{j_1, s_1, j_2, s_2} |j_1, \frac{3}{2} - s_1\rangle \langle j_2, \frac{3}{2} - s_2|. \quad (5.17)$$

Specifically, `(nmax+1)*rhoA[[j,s,j,s]]` is the probability of detecting the particle at \tilde{x}_j in spin state $\frac{3}{2} - s$. With this density matrix we calculate the following quantities:

Spin-specific densities: if we only detect particles in spin state $|\uparrow\rangle$, we measure in effect the expectation values of the spin-selective density operator $\hat{\rho}_\uparrow(\tilde{x}, \tilde{y}) = |\tilde{x}\rangle\langle\tilde{y}| \otimes |\uparrow\rangle\langle\uparrow|$. The spin-selective density values are found from the expansion (5.17) and the trace $\text{Tr}(\hat{A}) = \sum_{j=1}^{n_{\max}} \sum_{s=1}^2 \langle j, \frac{3}{2} - s | \hat{A} | j, \frac{3}{2} - s \rangle$:

$$\begin{aligned} \rho_\uparrow(\tilde{x}, \tilde{y}) &= \text{Tr}[\hat{\rho} \cdot \hat{\rho}_\uparrow(x, y)] \\ &= \text{Tr} \left[\sum_{j_1=1}^{n_{\max}} \sum_{s_1=1}^2 \sum_{j_2=1}^{n_{\max}} \sum_{s_2=1}^2 c_{j_1, s_1, j_2, s_2} |j_1, \frac{3}{2} - s_1\rangle \langle j_2, \frac{3}{2} - s_2| \cdot |\tilde{x}\rangle \langle \tilde{y}| \otimes |\uparrow\rangle \langle \uparrow| \right] \\ &= \sum_{j=1}^{n_{\max}} \sum_{s=1}^2 \sum_{j_1=1}^{n_{\max}} \sum_{s_1=1}^2 \sum_{j_2=1}^{n_{\max}} \sum_{s_2=1}^2 c_{j_1, s_1, j_2, s_2} \langle j, \frac{3}{2} - s | j_1, \frac{3}{2} - s_1 \rangle \langle j_2, \frac{3}{2} - s_2 | \tilde{x} \rangle \langle \tilde{y} | \uparrow \rangle \langle \uparrow | j, \frac{3}{2} - s \rangle \\ &= \sum_{j_1=1}^{n_{\max}} \sum_{j_2=1}^{n_{\max}} c_{j_1, \uparrow, j_2, \uparrow} \vartheta_{j_2}(\tilde{x}) \vartheta_{j_1}(\tilde{y}). \quad (5.18) \end{aligned}$$

Specifically, if $\tilde{x} = \tilde{x}_{j_x}$ and $\tilde{y} = \tilde{x}_{j_y}$ lie exactly on grid points of our finite-resolution calculation grid, then from Eq. (4.10) that

$$\rho_\uparrow(\tilde{x}_{j_x}, \tilde{x}_{j_y}) = (n_{\max} + 1) c_{j_x, \uparrow, j_y, \uparrow}. \quad (5.19)$$

That is, the detected density of spin-up particles is (at least on the grid points) given directly by the coefficients of `rhoA` computed above:

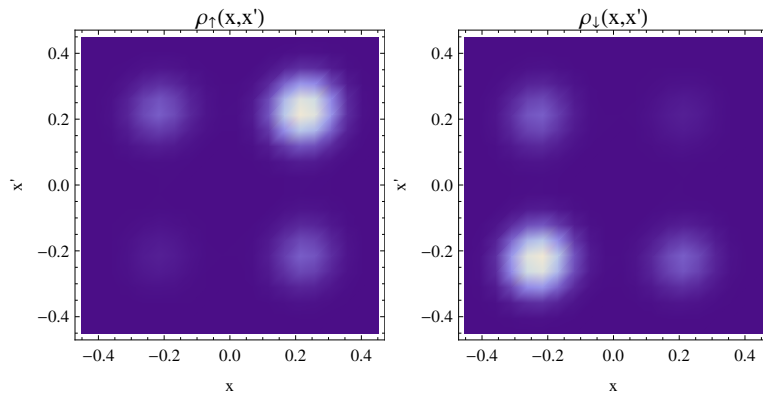
```
1 In[355]:= rhoup = (nmax+1) * rhoA[[All, 1, All, 1]];
```

In the same way the density of particles in the spin state $|\downarrow\rangle$ is

```
1 In[356]:= rhodown = (nmax+1) * rhoA[[All, 2, All, 2]];
```

They are plotted with

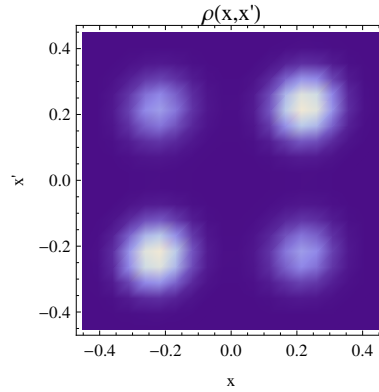
```
1 In[357]:= ListDensityPlot[rhoup, PlotRange -> All,
2           DataRange -> {{xval[[1]], xval[[-1]]},
3           {xval[[1]], xval[[-1]]}}]
```



Reduced density matrix of the spatial degree of freedom: using Eq. (3.32) we “trace out” the spin degree of freedom to find the density matrix in the spatial coordinate:

```
1 In[358]:=rhox = (nmax+1) * Sum[rhoA[[All,s,All s]], {s,1,2}];
```

This density is just the sum of the spin-specific densities shown above.

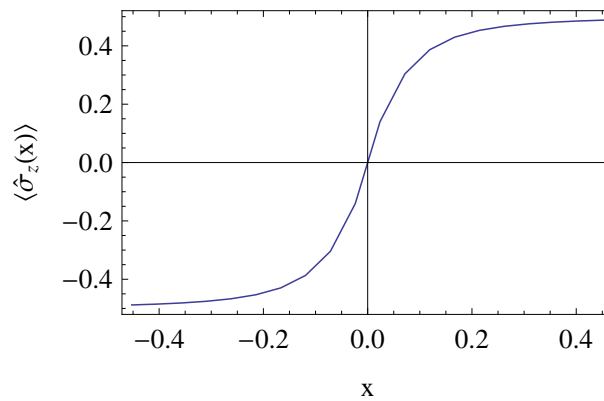


Reduced density matrix of the spin degree of freedom: we can do the same for the reduced matrix of the spin degree of freedom:

```
1 In[359]:=rhos = Sum[rhoA[[j, All, j, All]], {j, 1, nmax}]
2 Out[359]={{0.5, -0.205979}, {-0.205979, 0.5}}
```

Spin expectation value: if we only detect particles at a given position \tilde{x} , the expectation value for the measured spin is given by $\langle \hat{\sigma}_z(\tilde{x}) \rangle = \frac{+\frac{1}{2}\rho(\tilde{x}, \uparrow) - \frac{1}{2}\rho(\tilde{x}, \downarrow)}{\rho(\tilde{x}, \uparrow) + \rho(\tilde{x}, \downarrow)}$:

```
1 In[360]:=avgs = Table[Sum[rhoA[[j,s,j,s]]*(3/2-s), {s,1,2}]/
2 Sum[rhoA[[j,s,j,s]], {s,1,2}], {j,1,nmax}];
3 In[361]:=ListLinePlot[avgs, PlotRange -> All,
4 DataRange -> {xval[[1]], xval[[-1]]}]
```



This graph confirms the observation that particles detected on the left side are more likely to be in the $|\downarrow\rangle$ state, while particles detected on the right side are more likely to be in the $|\uparrow\rangle$ state.

5.1.3 exercises

Q5.1 In the problem described by the Hamiltonian of Eq. (5.5), calculate the following expectation values (numerically) for several parameter sets $\{\Omega, f, b_x\}$:

- $\langle \tilde{x} \rangle$ for particles detected in the $|\uparrow\rangle$ state
- $\langle \tilde{x} \rangle$ for particles detected in the $|\downarrow\rangle$ state
- $\langle \tilde{x} \rangle$ for particles detected in any spin state
- the mean and variance of $\hat{\tilde{x}} \otimes \hat{\sigma}_z$

Chapter 6

path-integral methods

With the approximations associated with the finite-resolution position basis set (section 4.1.1) we have significantly reduced the complexity of performing calculations of quantum-mechanical systems with continuous degrees of freedom such as their motion in space. When we study a very large number of particles, however, these approximations are still not sufficient and computers are still overwhelmed. Consider, for example, the extension of the problem of section 4.3.1 to N particles moving in three-dimensional space. Representing any wavefunction of this system in the position-basis requires n_{\max}^{3N} complex numbers; for $N = 20$ and $n_{\max} = 20$, which are both not very large numbers, we already require about 10^{78} complex numbers for the complete description, which approximates the number of particles in the universe.

The Trotter decomposition (4.31) we used in section 4.1.3 (real-time dynamics) and 4.2.1 (imaginary-time dynamics) lends itself to a quantum-mechanical description that circumvents this problem and can be used to estimate expectation values without calculating the full wavefunction or density matrix. And since we are ultimately interested in expectation values (measurable quantities), not in wavefunctions and density matrices (unmeasurable representations), this can be of significant use.

6.1 path integrals for propagation in time

Assume for a moment that we study a system with a time-independent Hamiltonian $\hat{\mathcal{H}}$. Equation (2.32) gives an explicit expression for the solution of the time-dependent Schrödinger equation through the propagator $\mathcal{U}(t) = \exp(-i\hat{\mathcal{H}}t/\hbar)$.

Here we wish to calculate matrix elements of this propagator in the position basis,

$$\langle \vec{x}' | e^{-i\hat{\mathcal{H}}t/\hbar} | \vec{x} \rangle, \quad (6.1)$$

where both \vec{x} and \vec{x}' are configuration vectors describing the $3N$ spatial coordinates of the system; $\vec{x} = \{x_1, y_1, z_1, x_2, y_2, z_2, \dots, x_N, y_N, z_N\}$. The matrix element (6.1) computes the amplitude with which a state (configuration) \vec{x} turns into a state (configuration) \vec{x}' during an evolution time t . It will be useful to look at the points \vec{x} and \vec{x}' as the starting and end points of a path through $3N$ -dimensional configuration space.

First we apply the Trotter expansion (4.31) to the propagator:

$$\begin{aligned}
 e^{-i\hat{\mathcal{H}}t/\hbar} &= \lim_{M \rightarrow \infty} \left(e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}} \right)^M = \lim_{M \rightarrow \infty} \left(e^{-\frac{it}{2M\hbar} \hat{\mathcal{H}}_{\text{pot}}} e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} e^{-\frac{it}{2M\hbar} \hat{\mathcal{H}}_{\text{pot}}} \right)^M \\
 &= \lim_{M \rightarrow \infty} e^{-\frac{it}{2M\hbar} \hat{\mathcal{H}}_{\text{pot}}} \underbrace{e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{pot}}} \dots e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{pot}}}}_{(M-1) \text{ repetitions of } e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{pot}}}} e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} e^{-\frac{it}{2M\hbar} \hat{\mathcal{H}}_{\text{pot}}}.
 \end{aligned} \tag{6.2}$$

Now we insert the unit operator

$$\begin{aligned}
 \mathbb{1} &= \int |\vec{x}\rangle \langle \vec{x}| d^{3N} \vec{x} \\
 &= \int_{-\infty}^{\infty} |x_1\rangle \langle x_1| dx_1 \otimes \int_{-\infty}^{\infty} |y_1\rangle \langle y_1| dy_1 \otimes \int_{-\infty}^{\infty} |z_1\rangle \langle z_1| dz_1 \otimes \dots \int_{-\infty}^{\infty} |z_N\rangle \langle z_N| dz_N
 \end{aligned} \tag{6.3}$$

between each two operators in (6.2). This unit operator integrates over all coordinates of all particles ($3N$ coordinates in total), and the vector \vec{x} represents all of these $3N$ coordinates; $|\vec{x}\rangle = |x_1\rangle \otimes |y_1\rangle \otimes |z_1\rangle \otimes \dots \otimes |z_N\rangle$. With these insertions, matrix elements of Eq. (6.2) become

$$\begin{aligned}
 \langle \vec{x}' | e^{-i\hat{\mathcal{H}}t/\hbar} | \vec{x} \rangle &= \lim_{M \rightarrow \infty} \int \langle \vec{x}'_M | e^{-\frac{it}{2M\hbar} \hat{\mathcal{H}}_{\text{pot}}} | \vec{x}_M \rangle \\
 &\times \underbrace{\langle \vec{x}_M | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} | \vec{x}'_{M-1} \rangle \langle \vec{x}'_{M-1} | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{pot}}} | \vec{x}_{M-1} \rangle \dots \langle \vec{x}_2 | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} | \vec{x}'_1 \rangle \langle \vec{x}'_1 | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{pot}}} | \vec{x}_1 \rangle}_{(M-1) \text{ repetitions of } \langle \vec{x}_{m+1} | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} | \vec{x}'_m \rangle \langle \vec{x}'_m | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{pot}}} | \vec{x}_m \rangle \text{ for } m = (M-1) \dots 1} \\
 &\times \langle \vec{x}_1 | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} | \vec{x}'_0 \rangle \langle \vec{x}'_0 | e^{-\frac{it}{2M\hbar} \hat{\mathcal{H}}_{\text{pot}}} | \vec{x}_0 \rangle d^{3N} \vec{x}'_0 d^{3N} \vec{x}_1 d^{3N} \vec{x}'_1 \dots d^{3N} \vec{x}'_{M-1} d^{3N} \vec{x}_M,
 \end{aligned} \tag{6.4}$$

where we have set $\vec{x}_0 = \vec{x}$ and $\vec{x}'_M = \vec{x}'$ as the starting and end points of the path. Equation (6.4) contains two kinds of integration variables, \vec{x}_m and \vec{x}'_m , which can be set equal because the potential Hamiltonian (including interaction potentials) is usually diagonal in the position representation: $\hat{\mathcal{H}}_{\text{pot}} = \int V(\vec{x}) |\vec{x}\rangle \langle \vec{x}| d^{3N} \vec{x}$, and therefore

$$\langle \vec{x}'_m | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{pot}}} | \vec{x}_m \rangle = \delta(\vec{x}_m - \vec{x}'_m) e^{-\frac{it}{M\hbar} V(\vec{x}_m)}. \tag{6.5}$$

Inserting Eq. (6.5) into (6.4) gives

$$\begin{aligned}
 \langle \vec{x}' | e^{-i\hat{\mathcal{H}}t/\hbar} | \vec{x} \rangle &= \lim_{M \rightarrow \infty} \int e^{-\frac{it}{2M\hbar} V(\vec{x}_M)} \\
 &\times \underbrace{\langle \vec{x}_M | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} | \vec{x}_{M-1} \rangle e^{-\frac{it}{M\hbar} V(\vec{x}_{M-1})} \dots \langle \vec{x}_2 | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} | \vec{x}_1 \rangle e^{-\frac{it}{M\hbar} V(\vec{x}_1)}}_{(M-1) \text{ repetitions of } \langle \vec{x}_{m+1} | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} | \vec{x}_m \rangle e^{-\frac{it}{M\hbar} V(\vec{x}_m)} \text{ for } m = (M-1) \dots 1} \\
 &\times \langle \vec{x}_1 | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} | \vec{x}_0 \rangle e^{-\frac{it}{2M\hbar} V(\vec{x}_0)} d^{3N} \vec{x}_1 \dots d^{3N} \vec{x}_{M-1},
 \end{aligned} \tag{6.6}$$

where $\vec{x}_0 = \vec{x}$ is the starting point and $\vec{x}_M = \vec{x}'$ is the end point of the path. A further simplification of Eq. (6.6) comes from the exact evaluation of the kinetic-energy matrix elements. If we assume that the kinetic energy represents the free motion of N particles of masses m_n in three dimensions,

$$\hat{\mathcal{H}}_{\text{kin}} = - \sum_{n=1}^N \frac{\hbar^2}{2m_n} \left(\frac{\partial^2}{\partial x_n^2} + \frac{\partial^2}{\partial y_n^2} + \frac{\partial^2}{\partial z_n^2} \right), \tag{6.7}$$

then the $3N$ coordinates of the particles propagate independently under $\hat{\mathcal{H}}_{\text{kin}}$. Therefore we first evaluate the matrix element for a single degree of freedom. With the conversion between the position basis $|x\rangle$ and the momentum basis $|k\rangle$ given by a Fourier transform,

$$|x\rangle = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dk e^{-ikx} |k\rangle \quad |k\rangle = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dx e^{ikx} |x\rangle \quad (6.8)$$

we find for a single coordinate (with $\alpha = \frac{\hbar t}{2mM}$)

$$\begin{aligned} \langle x' | e^{i\alpha \frac{\partial^2}{\partial x^2}} | x \rangle &= \langle x' | e^{i\alpha \frac{\partial^2}{\partial x^2}} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dk e^{-ikx} |k\rangle = \langle x' | \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dk e^{-i\alpha k^2} e^{-ikx} |k\rangle \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dk e^{-i\alpha k^2} e^{-ikx} \langle x' | k \rangle = \frac{1}{2\pi} \int_{-\infty}^{\infty} dk e^{-i\alpha k^2} e^{-ikx} e^{ikx'} = \sqrt{\frac{-i}{4\pi\alpha}} e^{\frac{i(x-x')^2}{4\alpha}}, \end{aligned} \quad (6.9)$$

since $e^{i\alpha \frac{\partial^2}{\partial x^2}} e^{ikx} = \left[\sum_{n=0}^{\infty} \frac{(i\alpha)^n}{n!} \frac{\partial^{2n}}{\partial x^{2n}} \right] e^{ikx} = \sum_{n=0}^{\infty} \frac{(i\alpha)^n}{n!} (ik)^{2n} e^{ikx} = \sum_{n=0}^{\infty} \frac{(-i\alpha k^2)^n}{n!} e^{ikx} = e^{-i\alpha k^2} e^{ikx}$. Therefore the full kinetic-energy propagator is

$$\langle \vec{x}' | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} | \vec{x} \rangle = \prod_{n=1}^N \left(\frac{-im_n M}{2\pi\hbar t} \right)^{\frac{3}{2}} \exp \left\{ \frac{im_n M [(x_n - x'_n)^2 + (y_n - y'_n)^2 + (z_n - z'_n)^2]}{2\hbar t} \right\}. \quad (6.10)$$

In the case where all particles have the same mass ($m_n = m \forall n$) this propagator can be simplified to

$$\langle \vec{x}' | e^{-\frac{it}{M\hbar} \hat{\mathcal{H}}_{\text{kin}}} | \vec{x} \rangle = \left(\frac{-imM}{2\pi\hbar t} \right)^{\frac{3N}{2}} \exp \left\{ \frac{imM \|\vec{x} - \vec{x}'\|^2}{2\hbar t} \right\}. \quad (6.11)$$

Using this latter form (6.11) for simplicity (a generalization to particles of unequal masses is straightforward but more complex), the expectation value (6.6) becomes

$$\begin{aligned} \langle \vec{x}' | e^{-i\hat{\mathcal{H}}t/\hbar} | \vec{x} \rangle &= \lim_{M \rightarrow \infty} \left(\frac{-imM}{2\pi\hbar t} \right)^{\frac{3NM}{2}} \int e^{-\frac{it}{2M\hbar} V(\vec{x}_M)} \\ &\times \underbrace{e^{\frac{imM}{2\hbar t} \|\vec{x}_M - \vec{x}_{M-1}\|^2} e^{-\frac{it}{M\hbar} V(\vec{x}_{M-1})} \dots e^{\frac{imM}{2\hbar t} \|\vec{x}_2 - \vec{x}_1\|^2} e^{-\frac{it}{M\hbar} V(\vec{x}_1)}}_{(M-1) \text{ repetitions of } e^{\frac{imM}{2\hbar t} \|\vec{x}_{m+1} - \vec{x}_m\|^2} e^{-\frac{it}{M\hbar} V(\vec{x}_m)} \text{ for } m = (M-1) \dots 1} \\ &\times e^{\frac{imM}{2\hbar t} \|\vec{x}_1 - \vec{x}_0\|^2} e^{-\frac{it}{2M\hbar} V(\vec{x}_0)} d^{3N} \vec{x}_1 \dots d^{3N} \vec{x}_{M-1}. \end{aligned} \quad (6.12)$$

Notice that in this form, the integrand does not contain any operators any more: it is simply a product of numbers. This means that we can gather them as a single exponential:

$$\begin{aligned} \langle \vec{x}' | e^{-i\hat{\mathcal{H}}t/\hbar} | \vec{x} \rangle &= \lim_{M \rightarrow \infty} \left(\frac{-imM}{2\pi\hbar t} \right)^{\frac{3NM}{2}} \int \exp \left[-\frac{it}{M\hbar} \left(\frac{1}{2} V(\vec{x}_0) + \sum_{m=1}^{M-1} V(\vec{x}_m) + \frac{1}{2} V(\vec{x}_M) \right) \right. \\ &\quad \left. + \frac{imM}{2\hbar t} \sum_{m=1}^M \|\vec{x}_m - \vec{x}_{m-1}\|^2 \right] d^{3N} \vec{x}_1 \dots d^{3N} \vec{x}_{M-1}. \end{aligned} \quad (6.13)$$

Now we look at the sequence $\vec{x}_0, \vec{x}_1, \dots, \vec{x}_M$ as a *path* through the space of configurations of our N -particle system. Since we are considering the limit $M \rightarrow \infty$, this path

will eventually become continuous. We let $\tilde{\mathbf{x}}(\tau)$ describe this continuous path, with $\tau = mt/M$ and $\tilde{\mathbf{x}}(mt/M) = \tilde{\mathbf{x}}_m$. The starting point is $\tilde{\mathbf{x}}(0) = \tilde{\mathbf{x}}$, and $\tilde{\mathbf{x}}(t) = \tilde{\mathbf{x}}'$ is the end point of our path. With this definition we can make the substitutions

$$\lim_{M \rightarrow \infty} \frac{t}{M} \left[\frac{1}{2} V(\tilde{\mathbf{x}}_0) + \sum_{m=1}^{M-1} V(\tilde{\mathbf{x}}_m) + \frac{1}{2} V(\tilde{\mathbf{x}}_M) \right] = \int_0^t V[\tilde{\mathbf{x}}(\tau)] d\tau \quad (6.14)$$

(trapezoidal integration) and

$$\lim_{M \rightarrow \infty} \frac{M}{t} \sum_{m=1}^M \|\tilde{\mathbf{x}}_m - \tilde{\mathbf{x}}_{m-1}\|^2 = \int_0^t \|\dot{\tilde{\mathbf{x}}}(\tau)\|^2 d\tau \quad (6.15)$$

(tacitly assuming that the path $\tilde{\mathbf{x}}(\tau)$ is differentiable). With these substitutions we re-write Eq. (6.13) as

$$\begin{aligned} & \langle \tilde{\mathbf{x}}' | e^{-i\hat{\mathcal{H}}t/\hbar} | \tilde{\mathbf{x}} \rangle \\ &= \lim_{M \rightarrow \infty} \left(\frac{-imM}{2\pi\hbar t} \right)^{\frac{3NM}{2}} \int \exp \left[\frac{i}{\hbar} \int_0^t \left(\frac{m}{2} \|\dot{\tilde{\mathbf{x}}}(\tau)\|^2 - V[\tilde{\mathbf{x}}(\tau)] \right) d\tau \right] d^{3N} \tilde{\mathbf{x}}_1 \cdots d^{3N} \tilde{\mathbf{x}}_{M-1}. \end{aligned} \quad (6.16)$$

We recognize the integrand in the exponential of Eq. (6.16) as the *Lagrangian*,

$$\mathcal{L}(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}) = \frac{m}{2} \|\dot{\tilde{\mathbf{x}}}\|^2 - V(\tilde{\mathbf{x}}), \quad (6.17)$$

and its integral as the *action* of the given path $\tilde{\mathbf{x}}(\cdot)$,

$$\mathcal{S}[\tilde{\mathbf{x}}(\cdot)] = \int_0^t \mathcal{L}(\tilde{\mathbf{x}}(\tau), \dot{\tilde{\mathbf{x}}}(\tau)) d\tau. \quad (6.18)$$

With this we find the final form of the matrix element of the propagator,

$$\langle \tilde{\mathbf{x}}' | e^{-i\hat{\mathcal{H}}t/\hbar} | \tilde{\mathbf{x}} \rangle = \int_{\tilde{\mathbf{x}}}^{\tilde{\mathbf{x}}'} e^{\frac{i}{\hbar} \mathcal{S}[\tilde{\mathbf{x}}(\cdot)]} \mathcal{D}_t[\tilde{\mathbf{x}}(\cdot)]. \quad (6.19)$$

The symbol $\int_{\tilde{\mathbf{x}}}^{\tilde{\mathbf{x}}'} \mathcal{D}_t[\tilde{\mathbf{x}}(\cdot)]$ denotes an integral over all (continuous and differentiable) paths $\tilde{\mathbf{x}}(\tau)$ running through $3N$ -dimensional configuration space from $\tilde{\mathbf{x}}(0) = \tilde{\mathbf{x}}$ to $\tilde{\mathbf{x}}(t) = \tilde{\mathbf{x}}'$. The pre-factor $\left(\frac{-imM}{2\pi\hbar t}\right)^{\frac{3NM}{2}}$ of Eq. (6.16) has been absorbed into this symbol, and in general path integrals as in Eq. (6.19) can only be interpreted up to a constant proportionality factor. When we calculate expectation values of operators, this is of no concern, as the following applications will show.

Consider now what we have achieved: starting from a quantum-mechanical expression for a matrix element (6.1), we have used the Trotter expansion to arrive at a numerical integral (6.19) which makes no reference to quantum mechanics at all. The price we pay is that Eq. (6.19) requires us to find *all* continuous and differentiable paths which take us from the initial configuration $\tilde{\mathbf{x}}$ to the final configuration $\tilde{\mathbf{x}}'$ within time t . In this sense, the quantum-mechanical propagation from one state to another goes through *all* possible paths simultaneously; the amplitudes of all these paths, given by their action integral, can interfere, as described by Eq. (6.19). This gives a very intuitive picture to experiments such as Young's double-slit, where the question of which slit the photon passes through is answered straightforwardly:

it passes through *both* slits, and the interference pattern results from the interference of the action integrals of the two paths.

While the path integral formula (6.19) looks clean and simple, it is not at all clear how such an integration is to be done in practice. For practical calculations, we return to Eq. (6.13) and use a finite number M of “time slices”. But if at each time slice we must integrate over the system’s spatial $3N$ coordinates (the $d^{3N}\tilde{\mathbf{x}}_1 \dots d^{3N}\tilde{\mathbf{x}}_{M-1}$ terms), this path integration is impossible for any reasonably-sized problem. So what have we gained? We have gained in that Eq. (6.13) is still a simple numerical integration (albeit with very many integration variables) and can therefore be *approximated* by powerful techniques for evaluating high-dimensional definite integrals. A commonly used technique is a stochastic *Monte-Carlo* evaluation of the path integral (the “Path-Integral Monte-Carlo” technique): instead of summing over *all* paths $\tilde{\mathbf{x}}(\tau)$ connecting the starting point with the end point, we try to randomly generate a *representative sample* of paths, for example with the Metropolis–Hastings algorithm.

6.2 path integrals for propagation in imaginary time

With the substitution $t \mapsto -i\hbar\beta = -i\hbar/(k_B T)$, as in section 4.2.1 (page 81), we can calculate matrix elements of the thermal density matrix from Eq. (6.13),

$$\langle \tilde{\mathbf{x}}' | e^{-\beta \hat{\mathcal{H}}} | \tilde{\mathbf{x}} \rangle = \lim_{M \rightarrow \infty} \left(\frac{mM}{2\pi\hbar^2\beta} \right)^{\frac{3NM}{2}} \int \exp \left[-\frac{\beta}{M} \left(\frac{1}{2} V(\tilde{\mathbf{x}}_0) + \sum_{m=1}^{M-1} V(\tilde{\mathbf{x}}_m) + \frac{1}{2} V(\tilde{\mathbf{x}}_M) \right) - \frac{mM}{2\hbar^2\beta} \sum_{m=1}^M \|\tilde{\mathbf{x}}_m - \tilde{\mathbf{x}}_{m-1}\|^2 \right] d^{3N}\tilde{\mathbf{x}}_1 \dots d^{3N}\tilde{\mathbf{x}}_{M-1}. \quad (6.20)$$

We again interpret the sequence $\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_M$ as a *path* through the space of configurations of our N -particle system. We let $\tilde{\mathbf{x}}(\tau)$ describe this continuous path, with $\tau = m\beta/M$ and $\tilde{\mathbf{x}}(m\beta/M) = \tilde{\mathbf{x}}_m$. The starting point is $\tilde{\mathbf{x}}(0) = \tilde{\mathbf{x}}$, and $\tilde{\mathbf{x}}(\beta) = \tilde{\mathbf{x}}'$ is the end point of our path. With this definition we can make the substitutions

$$\lim_{M \rightarrow \infty} \frac{\beta}{M} \left[\frac{1}{2} V(\tilde{\mathbf{x}}_0) + \sum_{m=1}^{M-1} V(\tilde{\mathbf{x}}_m) + \frac{1}{2} V(\tilde{\mathbf{x}}_M) \right] = \int_0^\beta V[\tilde{\mathbf{x}}(\tau)] d\tau \quad (6.21)$$

and

$$\lim_{M \rightarrow \infty} \frac{M}{\beta} \sum_{m=1}^M \|\tilde{\mathbf{x}}_m - \tilde{\mathbf{x}}_{m-1}\|^2 = \int_0^\beta \|\dot{\tilde{\mathbf{x}}}(\tau)\|^2 d\tau, \quad (6.22)$$

giving

$$\begin{aligned} & \langle \tilde{\mathbf{x}}' | e^{-\beta \hat{\mathcal{H}}} | \tilde{\mathbf{x}} \rangle \\ &= \lim_{M \rightarrow \infty} \left(\frac{mM}{2\pi\hbar^2\beta} \right)^{\frac{3NM}{2}} \int \exp \left[-\int_0^\beta \left(\frac{m}{2\hbar^2} \|\dot{\tilde{\mathbf{x}}}(\tau)\|^2 + V[\tilde{\mathbf{x}}(\tau)] \right) d\tau \right] d^{3N}\tilde{\mathbf{x}}_1 \dots d^{3N}\tilde{\mathbf{x}}_{M-1}. \end{aligned} \quad (6.23)$$

Notice the modified sign in the integrand, compared to Eq. (6.16).

6.2.1 example: a single particle in a 1D harmonic oscillator

As a first example we study the harmonic oscillator Hamiltonian

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m} \frac{d^2}{dx^2} + \frac{1}{2} m \omega^2 x^2. \quad (6.24)$$

In what follows we calculate thermal matrix elements $\langle x' | e^{-\beta \hat{\mathcal{H}}} | x \rangle$; the calculation of propagator matrix elements follows the same scheme.

summation over exact eigenstates

We can exactly diagonalize (6.24) with $\hat{\mathcal{H}}|n\rangle = E_n|n\rangle$, where the energies are $E_n = \hbar\omega(n + \frac{1}{2})$ and the eigenfunctions are $\langle x|n\rangle = \frac{H_n(x/\hat{x})}{\sqrt{2^n n! \hat{x} \sqrt{\pi}}} e^{-\frac{x^2}{2\hat{x}^2}}$ in terms of the Hermite polynomials $H_n(z)$ and with the length scale $\hat{x} = \sqrt{\frac{\hbar}{m\omega}}$. The exact thermal matrix elements are therefore

$$\begin{aligned} \langle x' | e^{-\beta \hat{\mathcal{H}}} | x \rangle &= \sum_{n,n'=0}^{\infty} \langle x' | n' \rangle \langle n' | e^{-\beta \hat{\mathcal{H}}} | n \rangle \langle n | x \rangle \\ &= \sum_{n=0}^{\infty} \langle x' | n \rangle e^{-\beta E_n} \langle n | x \rangle = \frac{1}{\hat{x} \sqrt{\pi}} \sum_{n=0}^{\infty} \frac{H_n(x/\hat{x}) H_n(x'/\hat{x})}{2^n n!} e^{-\frac{x^2+x'^2}{2\hat{x}^2}} e^{-\beta \hbar \omega (n + \frac{1}{2})} \\ &= \frac{1}{\hat{x} \sqrt{\pi}} e^{-\frac{x^2+x'^2}{2\hat{x}^2}} e^{-\frac{1}{2} \beta \hbar \omega} \sum_{n=0}^{\infty} \frac{H_n(x/\hat{x}) H_n(x'/\hat{x})}{n!} \left[\frac{1}{2} e^{-\beta \hbar \omega} \right]^n \\ &= \frac{\exp \left[-\left(\frac{x+x'}{2\hat{x}} \right)^2 \tanh \left(\frac{\zeta}{2} \right) - \left(\frac{x-x'}{2\hat{x}} \right)^2 \coth \left(\frac{\zeta}{2} \right) \right]}{\hat{x} \sqrt{2\pi \sinh(\zeta)}}, \quad (6.25) \end{aligned}$$

where we abbreviate $\zeta = \beta \hbar \omega = \frac{\hbar \omega}{k_B T}$ as the scaled inverse temperature, and we have made use of the identity

$$\sum_{n=0}^{\infty} \frac{H_n(x) H_n(y) w^n}{n!} = \frac{e^{\frac{2w(2w(x^2+y^2)-2xy)}{4w^2-1}}}{\sqrt{1-4w^2}}. \quad (6.26)$$

The partition function is

$$\begin{aligned} Z(\beta) &= \text{Tr}(e^{-\beta \hat{\mathcal{H}}}) = \int_{-\infty}^{\infty} \langle x | e^{-\beta \hat{\mathcal{H}}} | x \rangle dx = \int_{-\infty}^{\infty} \frac{\exp \left[-\left(\frac{x}{\hat{x}} \right)^2 \tanh \left(\frac{\zeta}{2} \right) \right]}{\hat{x} \sqrt{2\pi \sinh(\zeta)}} dx \\ &= \frac{1}{2} \text{csch} \left(\frac{\zeta}{2} \right) \quad (6.27) \end{aligned}$$

in terms of $\text{csch}(z) = 1/\sinh(z)$, and hence the density matrix is

$$\langle x' | \rho(\beta) | x \rangle = \frac{\langle x' | e^{-\beta \hat{\mathcal{H}}} | x \rangle}{\text{Tr}(e^{-\beta \hat{\mathcal{H}}})} = \frac{\exp \left[-\left(\frac{x+x'}{2\hat{x}} \right)^2 \tanh \left(\frac{\zeta}{2} \right) - \left(\frac{x-x'}{2\hat{x}} \right)^2 \coth \left(\frac{\zeta}{2} \right) \right]}{\hat{x} \sqrt{\pi \coth \left(\frac{\zeta}{2} \right)}}. \quad (6.28)$$

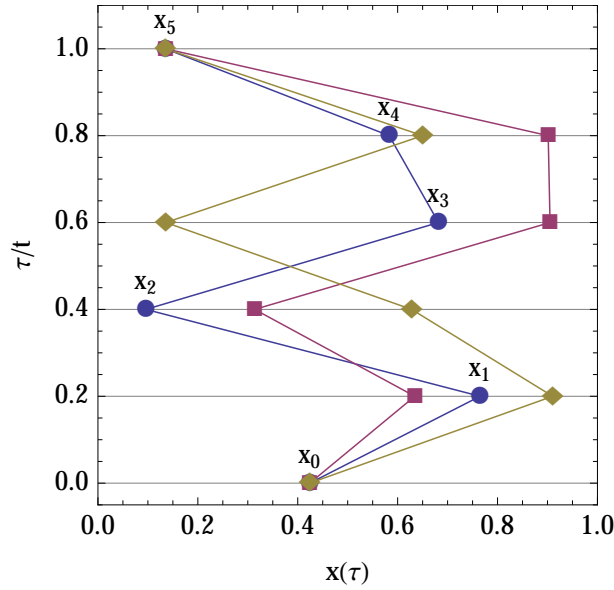
We will compare our path integral calculations with this exact result.

path integral formulation

In order to express the thermal density matrix as a path integral, we return to Eq. (6.20): for our one-dimensional problem ($3N \mapsto 1$),

$$\langle x' | e^{-\beta \hat{\mathcal{H}}} | x \rangle = \lim_{M \rightarrow \infty} \left(\frac{mM}{2\pi\hbar^2\beta} \right)^{\frac{M}{2}} \int_{-\infty}^{\infty} \exp \left[-\frac{m\omega^2\beta}{2M} \left(\frac{1}{2}x_0^2 + \sum_{m=1}^{M-1} x_m^2 + \frac{1}{2}x_M^2 \right) - \frac{mM}{2\hbar^2\beta} \sum_{m=1}^M (x_m - x_{m-1})^2 \right] dx_1 \cdots dx_{M-1}. \quad (6.29)$$

Here is an example of three different concrete paths for $M = 5$, starting at $x = x_0 = 0.42$ and ending at $x' = x_5 = 0.14$, passing through four intermediate points (x_1, x_2, x_3, x_4) :



For $M = 5$ we would now have to integrate over all possible intermediate points (x_1, x_2, x_3, x_4) , performing a four-dimensional integral, in order to find $\langle x' | e^{-\beta \hat{\mathcal{H}}} | x \rangle$.

Here we explicitly evaluate Eq. (6.29) for several values of M :

$M = 1$: The expression for the thermal density matrix elements becomes

$$\begin{aligned} \langle x' | \rho(\beta) | x \rangle &= \frac{\langle x' | e^{-\beta \hat{\mathcal{H}}} | x \rangle}{\text{Tr}(e^{-\beta \hat{\mathcal{H}}})} \\ &\approx \frac{\left(\frac{m}{2\pi\hbar^2\beta} \right)^{\frac{1}{2}} \exp \left[-\frac{m\omega^2\beta}{2} \left(\frac{1}{2}x^2 + \frac{1}{2}x'^2 \right) - \frac{m}{2\hbar^2\beta} (x' - x)^2 \right]}{\left(\frac{m}{2\pi\hbar^2\beta} \right)^{\frac{1}{2}} \int_{-\infty}^{\infty} \exp \left[-\frac{m\omega^2\beta}{2} \left(\frac{1}{2}\tilde{x}^2 + \frac{1}{2}\tilde{x}^2 \right) - \frac{m}{2\hbar^2\beta} (\tilde{x} - \tilde{x})^2 \right] d\tilde{x}} \\ &= \frac{1}{\hat{x}\sqrt{\pi}} \sqrt{\frac{\zeta}{2}} \exp \left[-\left(\frac{x+x'}{2\hat{x}} \right)^2 \frac{\zeta}{2} - \left(\frac{x-x'}{2\hat{x}} \right)^2 \frac{4+\zeta^2}{2\zeta} \right] \quad (6.30) \end{aligned}$$

where in the denominator we have set $x = x' = \tilde{x}$ in order to evaluate the trace. We notice that this expression matches Eq. (6.28) to first order in ζ , that is, Eq. (6.30) is a high-temperature approximation of Eq. (6.28).

$M = 2$: The expression for the thermal density matrix elements becomes

$$\begin{aligned} \langle x' | \rho(\beta) | x \rangle &= \frac{\langle x' | e^{-\beta \hat{\mathcal{H}}} | x \rangle}{\text{Tr}(e^{-\beta \hat{\mathcal{H}}})} \\ &\approx \frac{\left(\frac{m}{\pi \hbar^2 \beta} \right) \int_{-\infty}^{\infty} \exp \left[-\frac{m\omega^2 \beta}{4} \left(\frac{1}{2} x^2 + x_1^2 + \frac{1}{2} x'^2 \right) - \frac{m}{\hbar^2 \beta} [(x_1 - x)^2 + (x' - x_1)^2] \right] dx_1}{\left(\frac{m}{\pi \hbar^2 \beta} \right) \int_{-\infty}^{\infty} \exp \left[-\frac{m\omega^2 \beta}{4} \left(\frac{1}{2} \tilde{x}^2 + x_1^2 + \frac{1}{2} \tilde{x}^2 \right) - \frac{m}{\hbar^2 \beta} [(x_1 - \tilde{x})^2 + (\tilde{x} - x_1)^2] \right] dx_1 d\tilde{x}} \\ &= \frac{1}{\hat{x} \sqrt{\pi}} \sqrt{\frac{\zeta(16 + \zeta^2)}{4(8 + \zeta^2)}} \exp \left[-\left(\frac{x + x'}{2\hat{x}} \right)^2 \frac{\zeta}{4} \frac{16 + \zeta^2}{8 + \zeta^2} - \left(\frac{x - x'}{2\hat{x}} \right)^2 \frac{8 + \zeta^2}{4\zeta} \right]. \quad (6.31) \end{aligned}$$

This is a slightly better approximation of Eq. (6.28) for small ζ .

$M = 3$: The expression for the thermal density matrix elements becomes

$$\begin{aligned} \langle x' | \rho(\beta) | x \rangle &= \frac{\langle x' | e^{-\beta \hat{\mathcal{H}}} | x \rangle}{\text{Tr}(e^{-\beta \hat{\mathcal{H}}})} \\ &\approx \frac{\left(\frac{3m}{2\pi \hbar^2 \beta} \right)^{\frac{3}{2}} \int_{-\infty}^{\infty} \exp \left[-\frac{m\omega^2 \beta}{6} \left(\frac{1}{2} x^2 + x_1^2 + x_2^2 + \frac{1}{2} x'^2 \right) - \frac{3m}{2\hbar^2 \beta} [(x_1 - x)^2 + (x_2 - x_1)^2 + (x' - x_2)^2] \right] dx_1 dx_2}{\left(\frac{3m}{2\pi \hbar^2 \beta} \right)^{\frac{3}{2}} \int_{-\infty}^{\infty} \exp \left[-\frac{m\omega^2 \beta}{6} \left(\frac{1}{2} \tilde{x}^2 + x_1^2 + x_2^2 + \frac{1}{2} \tilde{x}^2 \right) - \frac{3m}{2\hbar^2 \beta} [(x_1 - \tilde{x})^2 + (x_2 - x_1)^2 + (\tilde{x} - x_2)^2] \right] dx_1 dx_2 d\tilde{x}} \\ &= \frac{1}{\hat{x} \sqrt{\pi}} \sqrt{\frac{243\zeta(16 + \zeta^2)}{32(9 + \zeta^2)(27 + \zeta^2)}} \exp \left[-\left(\frac{x + x'}{2\hat{x}} \right)^2 \frac{\zeta}{6} \frac{27 + \zeta^2}{9 + \zeta^2} - \left(\frac{x - x'}{2\hat{x}} \right)^2 \frac{(9 + \zeta^2)(36 + \zeta^2)}{6\zeta(27 + \zeta^2)} \right]. \quad (6.32) \end{aligned}$$

This is an even better approximation of Eq. (6.28) for small ζ .

$M \geq 4$: When you try to evaluate such integrals for a larger number M , in order to approach the limit $M \rightarrow \infty$, you will see that their evaluation takes a lot of computer power. They can be evaluated exactly in the present case of a harmonic oscillator; but in a more general case they cannot.

We see from this series of explicit calculations that taking a finite value for M yields a high-temperature approximation of the thermal density matrix (approximately correct for small ζ). The larger we choose M , the more the validity of the result extends to lower temperatures.

6.3 Monte Carlo integration

In Eqs. (6.13) and (6.23) we have expressed matrix elements of the real- and imaginary-time propagators as integrals over many-dimensional spaces [for N particles and M time-slices, there are $3N(M - 1)$ integration variables]. In this section we study a method for performing such integrals in practice, with a reasonable amount of computational power.

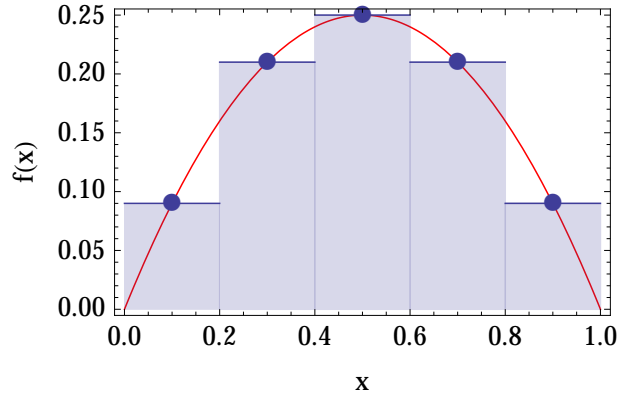
6.3.1 one-dimensional uniform integration

As a first example, we want to calculate a one-dimensional integral of the form

$$J = \int_0^1 f(x) dx, \quad (6.33)$$

where $f : [0, 1] \rightarrow \mathbb{C}$ is an arbitrary function.¹

Traditional Riemann integration of Eq. (6.33) can, for example, be done with the rectangular rule:



$$J = \lim_{M \rightarrow \infty} \sum_{m=1}^M \frac{1}{M} \times f\left(\frac{m-\frac{1}{2}}{M}\right) = \lim_{M \rightarrow \infty} \frac{\sum_{m=1}^M f\left(\frac{m-\frac{1}{2}}{M}\right)}{M}. \quad (6.34)$$

If the function $f(x)$ is sufficiently well-behaved, the sum over the regular x -grid in Eq. (6.34) can be replaced by a sum over a series of random numbers: if $(x_1, x_2, x_3, \dots, x_M)$ is a sequence of random numbers drawn independently and uniformly from $[0, 1]$, then $\langle f(x_m) \rangle = \int_0^1 f(x_m) dx_m$ and hence

$$\lim_{M \rightarrow \infty} \frac{\sum_{m=1}^M f(x_m)}{M} = \langle f(x_m) \rangle = J. \quad (6.35)$$

This formulation is called a *Monte-Carlo integral*, after the city of Monte Carlo famous for its gambling casinos.

In Mathematica, we first define the function $f(x)$, for example

```
1 In[362]:= f[x_] = x(1-x);
```

and the correct answer for the integral,

```
1 In[363]:= J = Integrate[f[x], {x, 0, 1}]
2 Out[363]= 1/6
```

Since `RandomReal[]` generates random numbers drawn uniformly from $[0, 1]$, we calculate an average of M random numbers with

```
1 In[364]:= Jmc[M_Integer/; M>=1] := Sum[f[RandomReal[]], {M}]/M
```

We can also simultaneously estimate the mean $J = \langle f(x) \rangle$ and its standard error $\sigma_J = \sqrt{\frac{\langle f^2(x) \rangle - \langle f(x) \rangle^2}{M}}$ from the same sequence (x_1, x_2, \dots, x_M) :

¹Choosing $[0, 1]$ as the domain of integration is arbitrary; we could have chosen any real finite or infinite interval.

```

1 In[365]:= Jmc[M_Integer;/;M>=1] := Module[{x,fx},
2   (* the values x_m *)
3   x = RandomReal[{0,1}, M];
4   (* the values f(x_m) *)
5   fx = f /@ x;
6   (* return mean and standard error *)
7   {Mean[fx], Sqrt[Variance[fx]/M]}]

```

For example, using $M = 10000$ we get a reasonable result for J :

```

1 In[366]:= Jmc[10000]
2 Out[366]= {0.166002, 0.000748734}

```

6.3.2 one-dimensional integration with weight

Next we wish to integrate a function $f : [0, 1] \rightarrow \mathbb{C}$ using a *weight function* $p : [0, 1] \rightarrow \mathbb{R}_0^+$ satisfying $\int_0^1 p(x) dx = 1$:

$$J = \int_0^1 f(x) p(x) dx \quad (6.36)$$

In principle, we could define $\tilde{f}(x) = f(x)p(x)$ and use the procedure of section 6.3.1. In practice, there is a much more efficient procedure: we define the *cumulative weight*

$$q(x) = \int_0^x p(y) dy, \quad (6.37)$$

which satisfies $q(0) = 0$, $q(1) = 1$, and is monotonically increasing and therefore uniquely invertible on $[0, 1]$, since $q'(x) = p(x) \geq 0$. Hence, using the variable substitution $z = q(x)$ we can re-express Eq. (6.36) as

$$J = \int_0^1 f[q^{-1}(z)] dz = \int_0^1 g(z) dz, \quad (6.38)$$

where we have defined $g(z) = f[q^{-1}(z)]$. Now we can use the procedure of section 6.3.1 on this function $g : [0, 1] \rightarrow \mathbb{C}$:

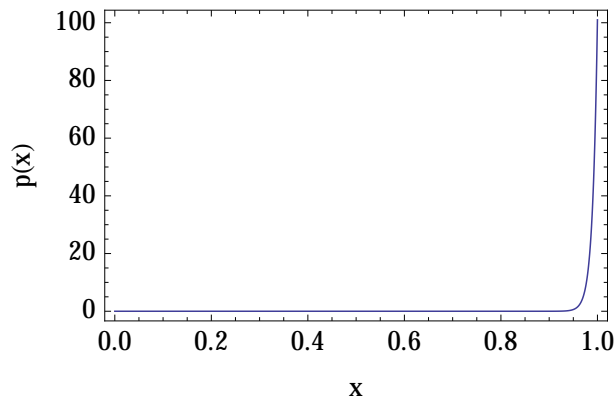
$$J = \lim_{M \rightarrow \infty} \frac{\sum_{m=1}^M g(z_m)}{M} = \lim_{M \rightarrow \infty} \frac{\sum_{m=1}^M f[q^{-1}(z_m)]}{M}, \quad (6.39)$$

where (z_1, z_2, \dots, z_M) is a sequence of random numbers drawn uniformly and independently from $[0, 1]$. We will show in an example that this is a much more efficient choice than using the $\tilde{f}(x)$ defined above.

Consider, for example, the weight function

$$p(x) = 101 \times x^{100} \quad (6.40)$$

which is sharply concentrated around $x = 1$ but remains nonzero throughout $(0, 1]$:



Defining the function $\tilde{f}(x) = f(x)p(x)$, as suggested above, means that when we use integration equation (6.35) on \tilde{f} then more than 90% of the random numbers x_m do not contribute significantly to the Monte Carlo estimate of J :

```

1 In[367]:=p[x_] = 101 * x^100;
2 In[368]:=J = Integrate[f[x]*p[x], {x,0,1}]
3 Out[368]=101/10506
4 In[369]:=Jmc1[M_Integer;/;M>=1] := Module[{x, fpx},
5     (* the values x_m *)
6     x = RandomReal[{0,1}, M];
7     (* the values f(x_m)*p(x_m) *)
8     fpx = f[#]p[#]& /@ x;
9     (* return mean and standard error *)
10    {Mean[fpx], Sqrt[Variance[fpx]/M]}]
11 In[370]:=Jmc1[10000]
12 Out[370]={0.00948913, 0.000480376}

```

We see that with 10 000 random numbers we got an estimate that has a relative precision of about 5%.

Now we calculate the cumulative weight

$$q(x) = \int_0^x p(y)dy = x^{101}, \quad (6.41)$$

which in this case we can invert to $q^{-1}(x) = x^{1/101}$. Using this, we calculate a second estimate of J :

```

1 In[371]:=q[x_] = Integrate[p[y], {y,0,x}]
2 Out[371]=x^101
3 In[372]:=Jmc2[M_Integer;/;M>=1] := Module[{z,x,fx},
4     (* the values z_m *)
5     z = RandomReal[{0,1}, M];
6     (* the values x_m = Inverse[q](z_m) *)
7     x = z^(1/101);
8     (* the values f(x_m) = g(z_m) *)
9     fx = f /@ x;
10    (* return mean and standard error *)
11    {Mean[fx], Sqrt[Variance[fx]/M]}]

```

```

12 In[373]:= Jmc2[10000]
13 Out[373]= {0.00951386, 0.0000907012}

```

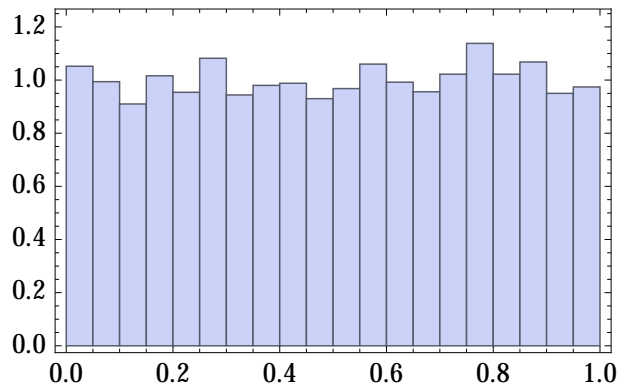
This time we get a relative precision of about 1% using the same number of random numbers.

Given a sequence of random numbers (z_1, z_2, \dots, z_M) drawn uniformly and independently from $[0, 1]$, we notice that the sequence (x_1, x_2, \dots, x_M) with $x_m = q^{-1}(z_m)$ is distributed according to p :

```

1 In[374]:= z = RandomReal[{0, 1}, 10000];
2 In[375]:= Histogram[z, Automatic, "PDF"]

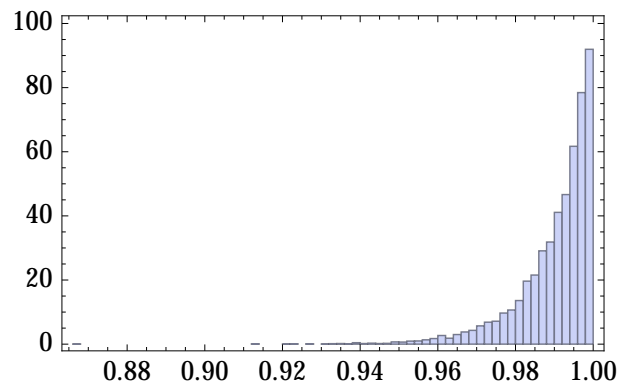
```



```

1 In[376]:= z = x^(1/101);
2 In[377]:= Histogram[x, Automatic, "PDF"]

```



Therefore, Eq. (6.39) calculates an average of $f(x)$ using random values of x_m drawn independently from the probability distribution $p(x)$.

6.3.3 the Metropolis–Hastings algorithm

In our specific example $p(x) = 101 \times x^{100}$, drawing random numbers from this distribution was relatively easy since the cumulative distribution $q(x) = x^{101}$ was analytically invertible. In more general cases, and in particular for multidimensional

probability distributions, this cannot be done and a different method for generating the random numbers x_m must be sought.

The simplest such algorithm is the Metropolis–Hastings algorithm, which generates a sequence of correlated random numbers (x_1, x_2, \dots, x_M) that are asymptotically ($M \rightarrow \infty$) distributed according to a probability density $p(x)$. It works as follows:

1. Start with a random starting value x_1 , and set $n = 1$.
2. Propose a candidate for x_{n+1} by drawing from a probability distribution $\pi(x_n \mapsto x_{n+1})$.
3. Calculate the acceptance ratio $P(x_n \mapsto x_{n+1}) = \min\left(1, \frac{p(x_{n+1})\pi(x_{n+1} \mapsto x_n)}{p(x_n)\pi(x_n \mapsto x_{n+1})}\right)$.
4. Choose a random number w_{n+1} from the uniform distribution over $[0, 1]$:
 - If $P(x_n \mapsto x_{n+1}) > w_{n+1}$, then we accept the move to x_{n+1} .
 - If $P(x_n \mapsto x_{n+1}) \leq w_{n+1}$, then we reject the move, and set $x_{n+1} = x_n$.
5. Increment n and go back to step 2.

Let's do an example: as a candidate distribution we use

$$\pi(x \mapsto x') = \begin{cases} \frac{1}{2\Delta} & \text{if } |x - x'| \leq \Delta, \\ 0 & \text{otherwise,} \end{cases} \quad (6.42)$$

which is symmetric and therefore $\frac{\pi(x_{n+1} \mapsto x_n)}{\pi(x_n \mapsto x_{n+1})} = 1$ simplifies the acceptance ratio $P(x_n \mapsto x_{n+1})$.

```

1 In[378]:=next[x_, d_] := Module[{y, P, w},
2   (* propose a new point *)
3   y = x + RandomReal[{-d,d}];
4   (* acceptance probability *)
5   P = If[y<0 || y>1, 0, Min[1, p[y]/p[x]]];
6   (* Metropolis-Hastings accept/reject *)
7   w = RandomReal[];
8   If[P > w, acc++; y, rej++; x]
9 In[379]:=MHchain[x1_?NumericQ, d_?NumericQ, M_Integer/;M>=1] :=
10   Module[{},
11     (* reset the acceptance/rejection counters *)
12     acc = rej = 0;
13     (* generate the chain of x values *)
14     NestList[next[#,d]&, x1, M-1]]

```

This procedure indeed generates a sequence (x_1, x_2, \dots, x_M) distributed according to $p(x)$ without making reference to the cumulative distribution $q(x)$ or its inverse $q^{-1}(x)$:

```

1 In[380]:=X = MHchain[1, 0.015, 10000];
2 In[381]:=acc/(acc + rej) // N
3 Out[381]=0.520152

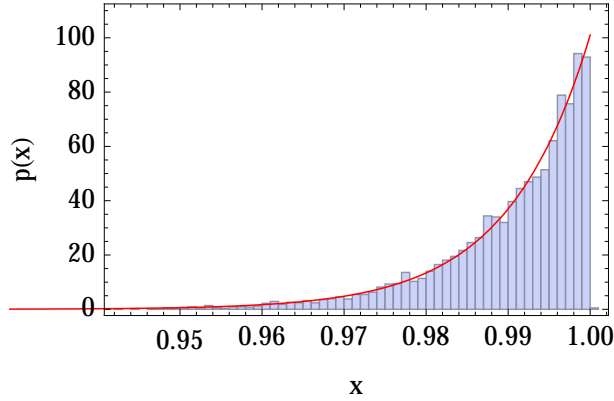
```

Notice that we have picked a step size $d = \Delta = 0.015$ such that the acceptance ratio is about 50%, i.e., about half of the proposed moves are accepted.

```

1 In[382]:=P1 = Plot[p[x], {x, 0, 1}, PlotRange -> All];
2 In[383]:=P2 = Histogram[X, Automatic, "PDF"];
3 In[384]:=Show[P2,P1]

```



How does this work? Assume that the Metropolis–Hastings algorithm ultimately (in the limit $M \rightarrow \infty$) generates a set of values x_m distributed according to a function $s(x)$. This distribution function $s(x)$ is therefore invariant under the Metropolis–Hastings algorithm, meaning that the detailed-balance condition $s(x)\pi(x \rightarrow x')P(x \rightarrow x') = s(x')\pi(x' \rightarrow x)P(x' \rightarrow x)$ must be satisfied. Inserting the definition of $P(x \rightarrow x')$,

$$s(x)\pi(x \rightarrow x') \min\left(1, \frac{p(x')\pi(x' \rightarrow x)}{p(x)\pi(x \rightarrow x')}\right) = s(x')\pi(x' \rightarrow x) \min\left(1, \frac{p(x)\pi(x \rightarrow x')}{p(x')\pi(x' \rightarrow x)}\right). \quad (6.43)$$

Since p and π are nonnegative, we can modify this to

$$\begin{aligned} s(x)\pi(x \rightarrow x') \frac{\min(p(x)\pi(x \rightarrow x'), p(x')\pi(x' \rightarrow x))}{p(x)\pi(x \rightarrow x')} \\ = s(x')\pi(x' \rightarrow x) \frac{\min(p(x')\pi(x' \rightarrow x), p(x)\pi(x \rightarrow x'))}{p(x')\pi(x' \rightarrow x)} \end{aligned} \quad (6.44)$$

and, noticing that the minimum on both sides of this equation is the same,

$$\frac{s(x)}{p(x)} = \frac{s(x')}{p(x')}. \quad (6.45)$$

The only way this equation can be satisfied for all (x, x') is if $s(x) \propto p(x)$. Since both $s(x)$ and $p(x)$ are normalized, we conclude that $s(x) = p(x)$: the stationary distribution of the Metropolis–Hastings algorithm is indeed $p(x)$, as desired.

What are such sequences (x_1, x_2, \dots, x_M) good for? Since we know that the points in such a sequence are distributed according to $p(x)$, we can now approximate integrals of the form (6.36) with

$$J = \int_0^1 f(x)p(x)dx = \lim_{M \rightarrow \infty} \frac{1}{M} \sum_{m=1}^M f(x_m). \quad (6.46)$$

6.4 Path-Integral Monte Carlo

We can now combine the multi-dimensional integrals of Eqs. (6.13) and (6.20) with the stochastic integration method of section 6.3.

We continue with the one-dimensional harmonic oscillator of section 6.2.1, in particular with Eq. (6.29) for the matrix elements of the thermal density matrix. Comparing Eq. (6.29) with Eq. (6.36), we identify the weight function

$$p(x_1, x_2, \dots, x_{M-1}) \propto \exp \left[-\frac{m\omega^2\beta}{2M} \left(\frac{1}{2}x_0^2 + \sum_{m=1}^{M-1} x_m^2 + \frac{1}{2}x_M^2 \right) - \frac{mM}{2\hbar^2\beta} \sum_{m=1}^M (x_m - x_{m-1})^2 \right]. \quad (6.47)$$

The goal of this section is to construct a sequence of paths whose elements are distributed according to this weight function: as shown in the example of [In \[384\]](#), the set shall contain more paths with high weight $p(x_1, x_2, \dots, x_M)$ and fewer paths with low weight. Notice that we need not be concerned with the pre-factor of p , as the Metropolis–Hastings algorithm will automatically find the correct normalization [see Eq. (6.45)].

The Metropolis–Hastings algorithm can now be set up to work in the space of *paths*, that is, in the space of vectors $\tilde{\mathbf{x}} = (x_1, x_2, \dots, x_{M-1})$, in the exact same way as we had set it up in section 6.3.3:

1. Start with a random starting path $\tilde{\mathbf{x}}^{(1)}$, and set $n = 1$. A useful starting point would be the path that interpolates linearly between the fixed end points x_0 and x_M , which is $x_m^{(1)} = x_0 + (m/M)(x_M - x_0)$.
2. Propose a candidate path $\tilde{\mathbf{x}}^{(n+1)}$ by drawing from a probability distribution $\pi(\tilde{\mathbf{x}}^{(n)} \mapsto \tilde{\mathbf{x}}^{(n+1)})$. There are many ways of proposing new paths, and the efficiency of the stochastic integration will depend strongly on the choices made at this point. The simplest choice for finding a candidate is to select a random index $\mu \in \{1, \dots, M-1\}$ and then add a random number Δx to $x_\mu^{(n)}$, so that $x_m^{(n+1)} = x_m^{(n)} + \delta_{m,\mu} \Delta x$.
3. Calculate the acceptance ratio $P(\tilde{\mathbf{x}}^{(n)} \mapsto \tilde{\mathbf{x}}^{(n+1)}) = \min \left(1, \frac{p(\tilde{\mathbf{x}}^{(n+1)})\pi(\tilde{\mathbf{x}}^{(n+1)} \mapsto \tilde{\mathbf{x}}^{(n)})}{p(\tilde{\mathbf{x}}^{(n)})\pi(\tilde{\mathbf{x}}^{(n)} \mapsto \tilde{\mathbf{x}}^{(n+1)})} \right)$. For the simple candidate mechanism above, the probability density is symmetric, $\pi(\tilde{\mathbf{x}}^{(n)} \mapsto \tilde{\mathbf{x}}^{(n+1)}) = \pi(\tilde{\mathbf{x}}^{(n+1)} \mapsto \tilde{\mathbf{x}}^{(n)})$, which simplifies the calculation of the acceptance ratio.
4. Choose a random number w_{n+1} from the uniform distribution over $[0, 1)$:
 - If $P(\tilde{\mathbf{x}}^{(n)} \mapsto \tilde{\mathbf{x}}^{(n+1)}) > w_{n+1}$, then we accept the move to $\tilde{\mathbf{x}}^{(n+1)}$.
 - If $P(\tilde{\mathbf{x}}^{(n)} \mapsto \tilde{\mathbf{x}}^{(n+1)}) \leq w_{n+1}$, then we reject the move, and set $\tilde{\mathbf{x}}^{(n+1)} = \tilde{\mathbf{x}}^{(n)}$.
5. Increment n and go back to step 2.

We notice that, in the form presented here, the algorithm generates a sample of paths from x_0 to x_M that approximates the desired path weight function (6.47), in the same way that in [In \[380\]](#) we had calculated a sample of values distributed according to the weight function given in Eq. (6.40); but it does not yet give us an estimate for the density matrix element $\langle x_M | e^{-\beta \hat{\mathcal{H}}} | x_0 \rangle$.

Let's set up such a calculation in Mathematica. To simplify the notation, the action of a path $\tilde{\mathbf{x}}$ at inverse temperature β is expressed in terms of the dimensionless

path coordinates $\tilde{x}_m = x_m / \hat{x}$ with the length scale $\hat{x} = \sqrt{\frac{\hbar}{m\omega}}$, and the dimensionless inverse temperature $\zeta = \beta \hbar \omega = \frac{\hbar \omega}{k_B T}$:

$$S(\tilde{\mathbf{x}}, \beta) = \frac{m\omega^2 \beta}{2M} \left(\frac{1}{2} x_0^2 + \sum_{m=1}^{M-1} x_m^2 + \frac{1}{2} x_M^2 \right) + \frac{mM}{2\hbar^2 \beta} \sum_{m=1}^M (x_m - x_{m-1})^2$$

$$= \frac{1}{2} \left[\frac{\zeta}{M} \left(\frac{1}{2} \tilde{x}_0^2 + \sum_{m=1}^{M-1} \tilde{x}_m^2 + \frac{1}{2} \tilde{x}_M^2 \right) + \frac{M}{\zeta} \sum_{m=1}^M (\tilde{x}_m - \tilde{x}_{m-1})^2 \right]. \quad (6.48)$$

With $\mathbf{x} = \tilde{\mathbf{x}} / \hat{x} = (\tilde{x}_0, \tilde{x}_1, \dots, \tilde{x}_M)$ and $\mathbf{z} = \zeta$ we calculate this action:

```

1 In[385]:=action[x_/_;VectorQ[x]&&Length[x]>=3, z_] :=
2       With[{M=Length[x]-1},
3           ((x[[1]]^2/2+Sum[x[[m]]^2,{m,2,M}]+x[[M+1]]^2/2)*(z/M)
4           + Sum[(x[[m+1]]-x[[m]])^2,{m,1,M}]*(M/z))/2]

```

Given a path \mathbf{x} , we find the next path via the Metropolis–Hastings algorithm, using a random step of size \mathbf{d} :

```

1 In[386]:=next[x_/_;VectorQ[x,NumericQ]&&Length[x]>=3,
2       z_?NumericQ, d_?NumericQ] :=
3       Module[{mu,dx,xn,S,Sn,P,w},
4           (* which point to modify *)
5           (* (leave end points fixed!) *)
6           mu = RandomInteger[{2,Length[x]-1}];
7           (* by how much to move the point *)
8           dx = RandomReal[{-d,d}];
9           (* the new path *)
10          xn = x; xn[[mu]] += dx;
11          (* calculate path actions *)
12          S = action[x,z];
13          Sn = action[xn,z];
14          (* acceptance probability *)
15          P = Min[1,Exp[S-Sn]];
16          (* acceptance or rejection *)
17          w = RandomReal[];
18          If[P > w, acc++; xn, rej++; x]]

```

The Path-Integral Monte Carlo (PIMC) algorithm for generating a sample of \mathbf{u} paths between $\tilde{x}_0 = \mathbf{x0}$ and $\tilde{x}_M = \mathbf{xM}$ taking $M = \mathbf{M}$ steps, at dimensionless inverse temperature $\zeta = \mathbf{z}$, taking a random step $\Delta x = \mathbf{d}$ on average, looks thus:

```

1 In[387]:=PIMCpaths[{x0_?NumericQ, xM_?NumericQ},
2       M_Integer;/M>=2, z_?NumericQ, d_?NumericQ,
3       u_Integer;/u>=1] :=
4       Module[{x},
5           (* start with the straight path *)
6           x = x0 + Range[0,M]/M * (xM-x0);
7           (* reset acceptance/rejection counters *)
8           acc = rej = 0;

```

```

9      (* iterate the Metropolis-Hastings algorithm *)
10     NestList[next[#,z,d]&, x, u]]

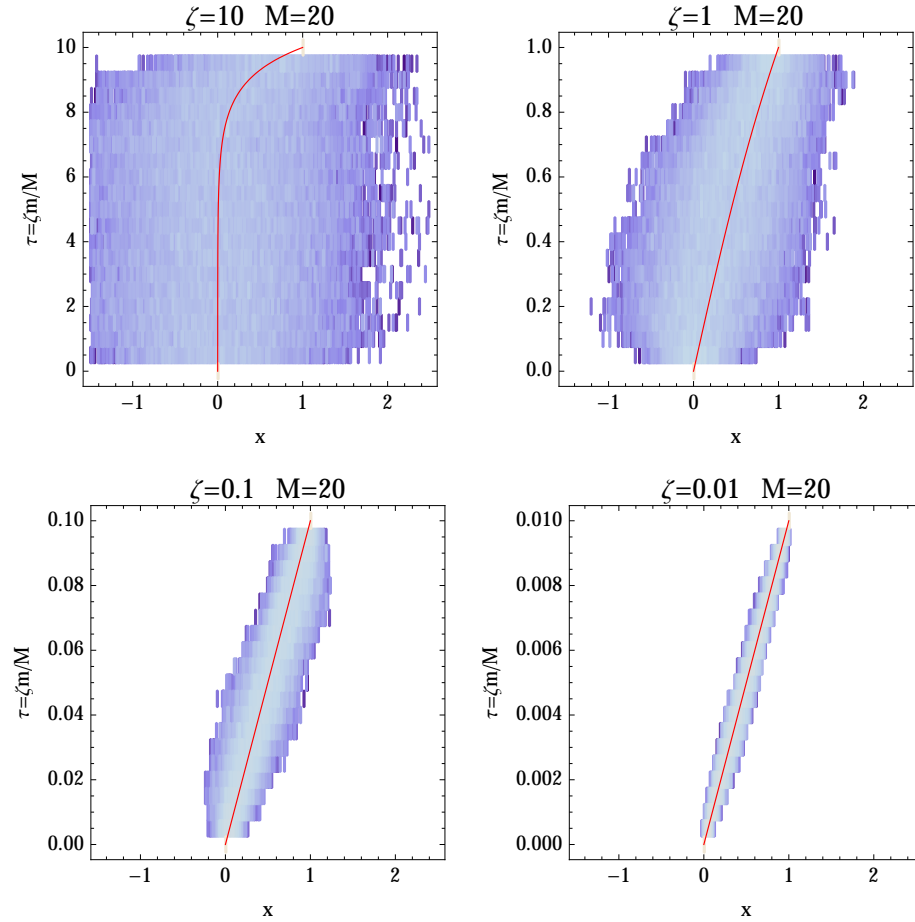
```

Here is a graphical representation of 10^5 paths between $\tilde{x}_0 = 0$ and $\tilde{x}_M = 1$ using $M = 20$ imaginary-time slices, at a dimensionless inverse temperatures of $\zeta = 10, 1, 0.1, 0.01$:

```

1  In[388]:=With[{z=1, M=20, d=0.5, u=10^5},
2      t = z * Range[0, M]/M;
3      p = PIMCpaths[{0, 1}, M, z, d, u];
4      DensityHistogram[Flatten[Transpose[{#,t}]&/@p,1],
5          {Automatic, {-(z/(2M)), z(1+1/(2M)), z/M}},
6          {"Log", "PDF"}]]

```



For smaller values of ζ , corresponding to higher temperatures, the paths are more and more concentrated around the straight path (the “least action” path of classical mechanics, indicated in red).

We notice that the output of `PIMCpaths`, which is a sequence of paths, does not directly allow us to calculate the matrix element $\langle x' | e^{-\beta \hat{\mathcal{H}}} | x \rangle$ from Eq. (6.29); instead, we can only evaluate integrals of the form of Eq. (6.46). In what follows, we will see what expectation values we can calculate directly from such path sequences.

6.4.1 calculating the density

The first observable quantity we wish to calculate is the thermal particle density

$$\rho(\vec{x}) = \langle \vec{x} | \hat{\rho} | \vec{x} \rangle = \frac{\langle \vec{x} | e^{-\beta \hat{\mathcal{H}}} | \vec{x} \rangle}{\int \langle \vec{x}' | e^{-\beta \hat{\mathcal{H}}} | \vec{x}' \rangle d^{3N} \vec{x}'}. \quad (6.49)$$

Both the numerator and the denominator of this expression are diagonal matrix elements of $e^{-\beta \hat{\mathcal{H}}}$ and can be written as *closed* path integrals, where the end point is equal to the starting point of the paths: using Eq. (6.20) with $\vec{x}_0 = \vec{x}_M = \vec{x}$ and $\vec{x}'_0 = \vec{x}'_M = \vec{x}'$,

$$\rho(\vec{x}) = \lim_{M \rightarrow \infty} \frac{\int \exp \left[-\frac{\beta}{M} \left(\frac{1}{2} V(\vec{x}_0) + \sum_{m=1}^{M-1} V(\vec{x}_m) + \frac{1}{2} V(\vec{x}_M) \right) - \frac{mM}{2\hbar^2 \beta} \sum_{m=1}^M \|\vec{x}_m - \vec{x}_{m-1}\|^2 \right] d^{3N} \vec{x}_1 \cdots d^{3N} \vec{x}_{M-1}}{\int \exp \left[-\frac{\beta}{M} \left(\frac{1}{2} V(\vec{x}'_0) + \sum_{m=1}^{M-1} V(\vec{x}'_m) + \frac{1}{2} V(\vec{x}'_M) \right) - \frac{mM}{2\hbar^2 \beta} \sum_{m=1}^M \|\vec{x}'_m - \vec{x}'_{m-1}\|^2 \right] d^{3N} \vec{x}'_0 d^{3N} \vec{x}'_1 \cdots d^{3N} \vec{x}'_M} \quad (6.50)$$

Notice that the denominator contains one more integration variable, $d^{3N} \vec{x}'_0$, as required in Eq. (6.49). Assume now that we have an infinite sequence of closed paths ($\vec{x}_M = \vec{x}_0$) through $3N$ -dimensional configuration space, with asymptotic distribution proportional to

$$p(\vec{x}_0, \vec{x}_1, \dots, \vec{x}_{M-1}) \propto \exp \left[-\frac{\beta}{M} \left(\frac{1}{2} V(\vec{x}_0) + \sum_{m=1}^{M-1} V(\vec{x}_m) + \frac{1}{2} V(\vec{x}_M) \right) - \frac{mM}{2\hbar^2 \beta} \sum_{m=1}^M \|\vec{x}_m - \vec{x}_{m-1}\|^2 \right]. \quad (6.51)$$

Of all these paths, the numerator of Eq. (6.50) contains only those that start and end at \vec{x} , while the denominator contains all of the paths:

$$\rho(\vec{x}) = \frac{\text{number of closed paths starting from and ending in } \vec{x}}{\text{number of closed paths}}. \quad (6.52)$$

We notice further that since these paths are closed, we cannot tell which point is their starting point and which is their end point; this insight improves the statistics of Eq. (6.52) by a factor of M to

$$\rho(\vec{x}) = \frac{\text{number of closed paths containing } \vec{x}}{M \text{ times the number of closed paths}}. \quad (6.53)$$

In practice, calculating the density in $3N$ -dimensional configuration space thus boils down to making a $3N$ -dimensional histogram of all the points contained in all the closed paths of the sequence. We will illustrate this with our harmonic oscillator example.

thermal density of a harmonic oscillator

The thermal density of a harmonic oscillator can be calculated analytically from Eq. (6.28):

$$\rho(x) = \langle x | \rho(\beta) | x \rangle = \frac{\exp \left[-\left(\frac{x}{\hat{x}} \right)^2 \tanh \left(\frac{\zeta}{2} \right) \right]}{\hat{x} \sqrt{\pi \coth \left(\frac{\zeta}{2} \right)}}. \quad (6.54)$$

We calculate a sequence of *closed* paths, where the beginning/end of the path is mobile as well, through a slight modification of [In \[386\]](#) and [In \[387\]](#): the last element

of the list $\mathbf{x} = (x_0, x_1, \dots, x_{M-1})$ has been chopped off, since it is identical to the first element. However, if we re-use the code of [In\[386\]](#) with this modification, we notice quickly that the convergence of the path sequence to the desired distribution is terribly slow. The reason is apparent: if we move only one point of the path at a time, it takes a long time until the entire path can move to a different place. The scale of motion of a single point on the path is given by the thermal de Broglie wavelength of the particle, and therefore goes to zero at high temperature; at the same time, the scale of motion of the entire path is given by the thermal width of the density, which becomes larger at high temperature. We must therefore introduce a second type of move, one that displaces the entire ring.

The first type of move remains the same: displace one point on the path by a random distance.

```

1 In[389]:=nextC1[x_/;VectorQ[x,NumericQ]&&Length[x]>=2,
2           z_?NumericQ, d_?NumericQ] :=
3           Module[{mu,dx,xn,S,Sn,P,w},
4             (* which point to modify *)
5             mu = RandomInteger[{1, Length[x]}];
6             (* by how much to move the point *)
7             dx = RandomReal[{-d, d}];
8             (* the new path *)
9             xn = x; xn[[mu]] += dx;
10            (* calculate path actions *)
11            S = action[Append[x, First[x]], z];
12            Sn = action[Append[xn, First[xn]], z];
13            (* acceptance probability *)
14            P = Min[1, Exp[S-Sn]];
15            (* acceptance or rejection *)
16            w = RandomReal[];
17            If[P>w, acc1++;xn, rej1++;x]]

```

The second type of move displaces the entire path (ring) by a random distance:

```

1 In[390]:=nextC2[x_/;VectorQ[x,NumericQ]&&Length[x]>=2,
2           z_?NumericQ, d_?NumericQ] :=
3           Module[{dx,xn,S,Sn,P,w},
4             (* by how much to move the points *)
5             dx = RandomReal[{-d, d}];
6             (* the new path *)
7             xn = x + dx;
8             (* calculate path actions *)
9             S = action[Append[x, First[x]], z];
10            Sn = action[Append[xn, First[xn]], z];
11            (* acceptance probability *)
12            P = Min[1, Exp[S-Sn]];
13            (* acceptance or rejection *)
14            w = RandomReal[];
15            If[P>w, acc2++;xn, rej2++;x]]

```

At every iteration, we choose a move of type 1 with probability f and a move of type 2 with probability $1 - f$:

```

1 In[391]:=nextC[x_/;VectorQ[x,NumericQ]&&Length[x]>=2,
2           z_?NumericQ, d1_?NumericQ, d2_?NumericQ,
3           f_?NumericQ] :=
4           If[RandomReal[]>f, nextC2[x,z,d2], nextC1[x,z,d1]]

```

Construct a sequence of closed paths:

```

1 In[392]:=PIMCpathsC[x0_?NumericQ,
2             M_Integer;/;M>=2, z_?NumericQ, d1_?NumericQ,
3             d2_?NumericQ, f_?NumericQ, u_Integer;/;u>=1] :=
4             Module[{x},
5               (* start with the trivial path at x0 *)
6               x = Table[x0, {M}];
7               (* reset acceptance/rejection counters *)
8               acc1 = rej1 = acc2 = rej2 = 0;
9               (* iterate the Metropolis-Hastings algorithm *)
10              NestList[nextC[#,z,d1,d2,f]&, x, u]]

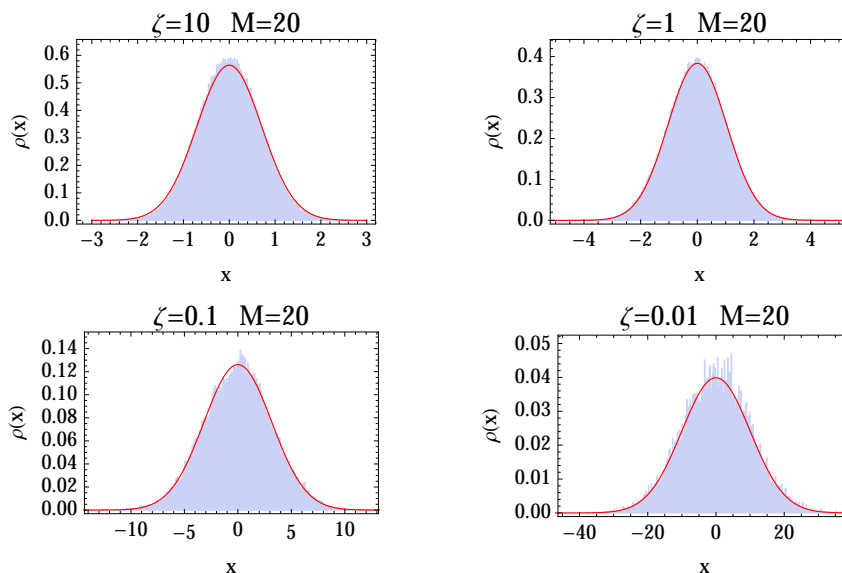
```

The density is found by plotting a histogram of all the points contained in all the paths:

```

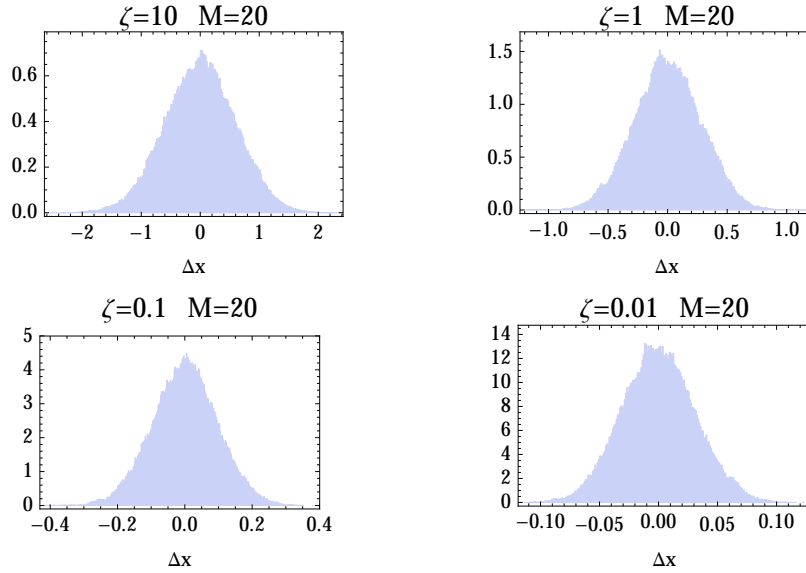
1 In[393]:=With[{z=1, M=20, d1=0.45, d2=3, f=0.5, u=10^5},
2           X = PIMCpathsC[0, M, z, d1, d2, f, u];
3           Histogram[Flatten[X], Automatic, "PDF"]

```



We see that these densities match the analytic expressions [red lines, Eq. (6.54)] within statistical uncertainties. While the spread of each path decreases with decreasing ζ (see In[388]), the overall size of the density profile *increases* with decreasing ζ ; hence the need for two different kinds of random moves above. We can see this decreasing ring size by moving each ring such that its center of gravity is at $x = 0$, and plotting a histogram of the resulting points:


```
1 In[394]:=Histogram[Flatten[#-Mean[#]&/@X], Automatic, "PDF"]
```



In the classical limit ($\zeta \rightarrow 0$), the closed paths are reduced to loops of zero length, *i.e.*, points, but these points are distributed over a large region in space. This is a simple example of how path integrals provide an intuitive picture of the classical limit of quantum mechanics.

Index

- action, 104
- angular momentum, 41
- Arnoldi algorithm, 25
- basis set, 31, 38
 - construction, 37
 - finite-resolution position basis, 66
 - incomplete, 32
 - momentum basis, 66
 - position basis, 65
- Bohr magneton, 43, 45
- Boltzmann constant, 81
- Bose–Einstein condensate, 80, 88
- C, 14, 22
- chemical potential, 83
- Clebsch–Gordan coefficient, 10, 53
- completeness relation, 31, 32, 65
- Confucius, 1
- contact interaction, 84
- correlations, 60
- Coulomb interaction, 87
 - truncated, 87
- detuning, 52
- Dicke states, 37, 38, 41
- discrete sine transform, 69, 89
- electron, 43
- energy gap, 57
- entanglement, 61
- entropy of entanglement, 61
- fast Fourier transform, 69
- Fortran, 22
- g*-factor, 43, 45
- Gross-Pitaevskii equation, *see* Schrödinger equation, non-linear
- ground state, 81
- harmonic oscillator, 38, 106
- harmonic well, 88
- Heisenberg model, 64
- Hilbert space, 31, 32, 34, 37, 44, 57, 65, 93
- hydrogen, 38
- hyperfine interaction, 45
- imaginary-time propagation, 81, 105
- interaction, 38, 84
- interaction picture, 36
- Ising model, 54, 63
- Java, 14, 22
- kinetic energy, 38, 66, 69, 103
- Lagrangian, 104
- Lanczos algorithm, 25
- level shift, 53
- magnetic field, 43, 44
- magnetization, 59
- Magnus expansion, 35
- Mandelbrot set, 10
- Mathematica, 9
 - anonymous function, 15, 20
 - assumptions, 27
 - brackets, 12, 22, 23
 - complex number, 27
 - conditional execution, 14
 - delayed assignment, 12, 17
 - differential equation, 50
 - factorial, 19
 - fixed point of a map, 83
 - front end, 9
 - function, 12, 16
 - functional programming, 15, 20, 21, 89
 - immediate assignment, 11, 16
 - kernel, 9
 - Kronecker product, 39, 40, 45, 55, 85
 - list, 13, 22
 - loop, 14, 20

- matrix, 23, 32
 - eigenvalues, 24, 44, 46
 - eigenvectors, 25, 44, 46
 - exponential, 36, 37
 - identity matrix, 40, 42
 - matrix exponential, 77
 - printing, 13, 23, 42
 - sparse matrix, 23, 42
- minimization, 49
- module, 14
- nesting function calls, 79
- numerical evaluation, 13
- outer product, 89
- pattern, 16, 18, 21, 24, 42
- physical units, 28, 43, 45
- piecewise functions, 72
- plotting, 46, 51, 72, 85
- postfix notation, 13
- prefix notation, 13
- procedure, *see* function
- random number, 11, 17
- recursion, 20, *see also* recursion
- remembering results, 18
- replacement, 72
- root finding, 71
- timing a calculation, 13, 42
- units, *see* physical units
- variable, 11
- vector, 22, 32
 - normalize, 82
 - orthogonal vectors, 46
- why?, 8
- Matlab, 22
- matrix-product state, 57
- mean-field interaction, 80
- Metropolis–Hastings algorithm, 105, 112
- Monte Carlo integration, 105
- Moore’s law, 57
- nuclear spin, 45
- operator, 31, 39
- oscillating field, 49
- partial trace, 62, 98
- path integral, 57, 101
- Pauli matrices, 32, 42, 43
- Planck’s constant, 43, 45, 50, 88
- plane wave, 38
- potential energy, 38
- product state, 40, 55
- propagator, 34, 76, 101
- Python, 14, 22
- quantum chemistry, 8
- quantum computing, 8
- quantum phase transition, 58
- quantum state, 31, 39
- Rabi frequency, 52
- real-space dynamics, 65, 93
- reduced density matrix, *see* partial trace
- rotating-wave approximation, 52
- rotation, 42
- Rubidium-87, 44, 88
 - magic field, 48
- s-wave scattering, 80, 84, 88
- Schrödinger equation
 - non-linear, 80, 83
 - time-dependent, 34, 36, 50
 - time-independent, 33, 43
- spherical harmonics, 38
- spin, 38, 41, 44, 93
- split-step method, 69, 79, 80
- square well, 38, 70
- Sturm–Liouville theorem, 66
- tensor product, 38, 45, 55, 84, 93, 97
- Thomas–Fermi approximation, 91
- transition matrix elements, 50
- Trotter expansion, 77, 79, 101
- von Neumann entropy, 62
- XY model, 64
- Zeeman shift
 - ac, 53
 - dc, 47